

RasPi

DESIGN
BUILD
CODE

34

Get hands-on with your Raspberry Pi



CODE AN
EMAIL
MONITOR

MASTER PIXEL DESKTOP

Plus Program and play a game for Sense HAT



Welcome



Raspbian has had its biggest update ever thanks to a dazzling new desktop environment called Pixel. This huge software

update gives you a desktop environment that wouldn't look out of place on a personal computer, rather than a microcomputer like the Raspberry Pi. Read on to discover how to use Pixel's crisp new interface and all of its new features and programs.

If you're going to start using Pixel as your default desktop, you might also want to create an email monitor to alert you to incoming messages using Python. If you prefer to get hands on with your Pi projects, you can also discover a Wi-Fi-enable walkie-talkie, how to hack your toys and bring them to life, as well as make a game with Sense HAT.

Jack Parsons
Editor

From the makers of
Linux User
& Developer

Join the conversation at...



@linuxusermag



Linux User & Developer



linuxuser@futurenet.com

Get inspired

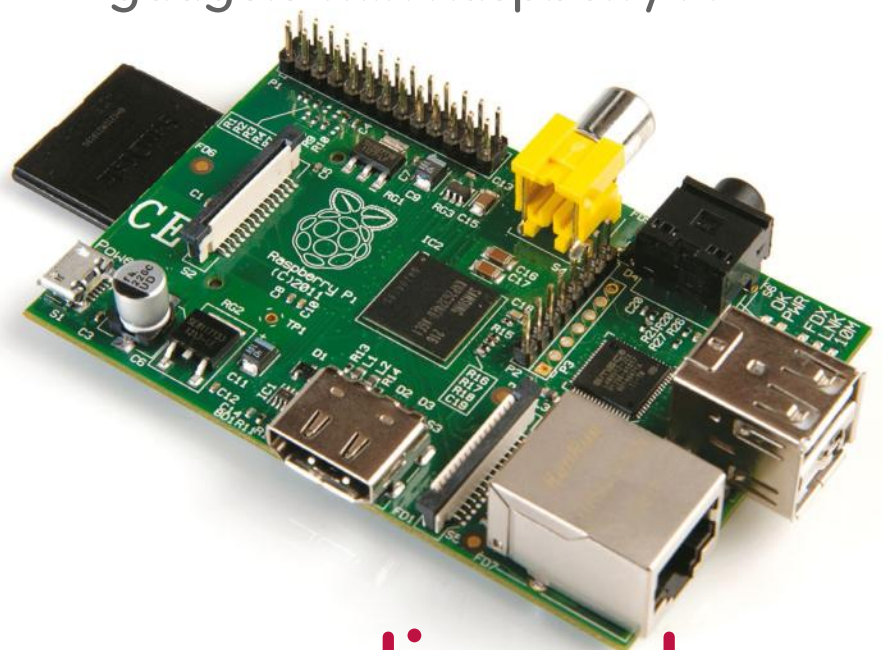
Discover the RasPi community's best projects

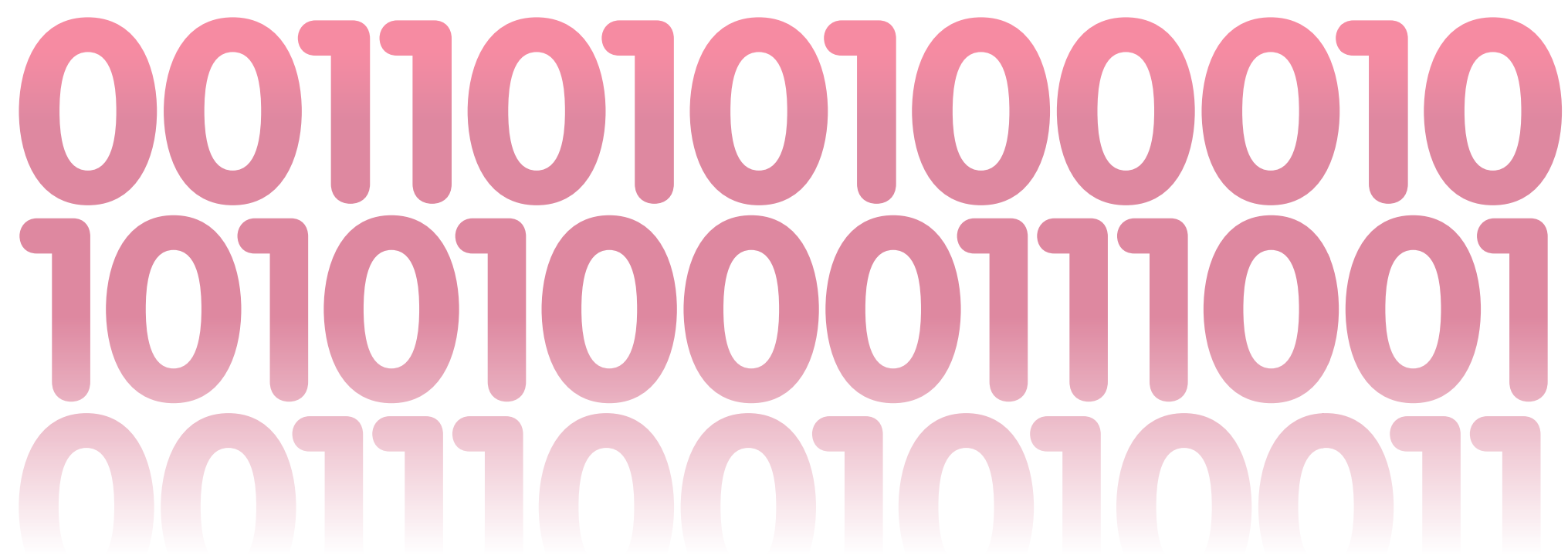
Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi





Contents

Master Pixel desktop

Get to grip with Raspberry Pi's official Raspbian replacement



Pi project: TalkiePi

Build your own Wi-Fi walkie-talkie with Raspberry Pi



Make an egg-drop game with the Sense HAT

Code your own game with out of this world tech



Hack a toy: Part 1

Learn four simple hacks to bring your toys to life



Check your email

Use Python to monitor your incoming messages

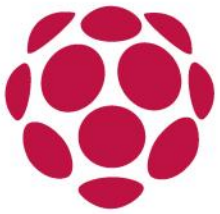


Talking Pi

Your questions answered and your opinions shared







Pixel (Pi Improved Xwindows Environment, Lightweight), is the latest iteration of the Raspbian desktop. The major changes are apparent the first time you boot up – the multitude of boot messages has been replaced with a simple splash screen with the release number.

There are now 16 stunning desktop background images to choose from thanks to Pi Foundation developer Greg Annandale. The icons on the file manager, task bar and menu now have a crisp, professional appearance. Menus are also cleaner and more readable as application icons no longer appear by default.

The rather clunky windows we formerly knew in Raspbian have now been replaced with rounded corners and a modified title bar. The infinality patchset also makes for much cleaner font rendering.

RealVNC Server is now bundled to allow you to easily select VNC from the interfaces menu, then connect via a viewer. There's also a provisional release of Chromium for the Pi, which in combination with the h264ify plugin makes use of the Pi's hardware to stream video.



THE PROJECT ESSENTIALS

Raspberry Pi (Ideally a Pi 2 or 3 if you want to use Chromium for streaming video)

If installing from Raspbian from scratch, an 8GB Micro SD card

Access to a computer for connecting via SSH and/or to install Raspbian Jessie with Pixel

01 Choose your install method

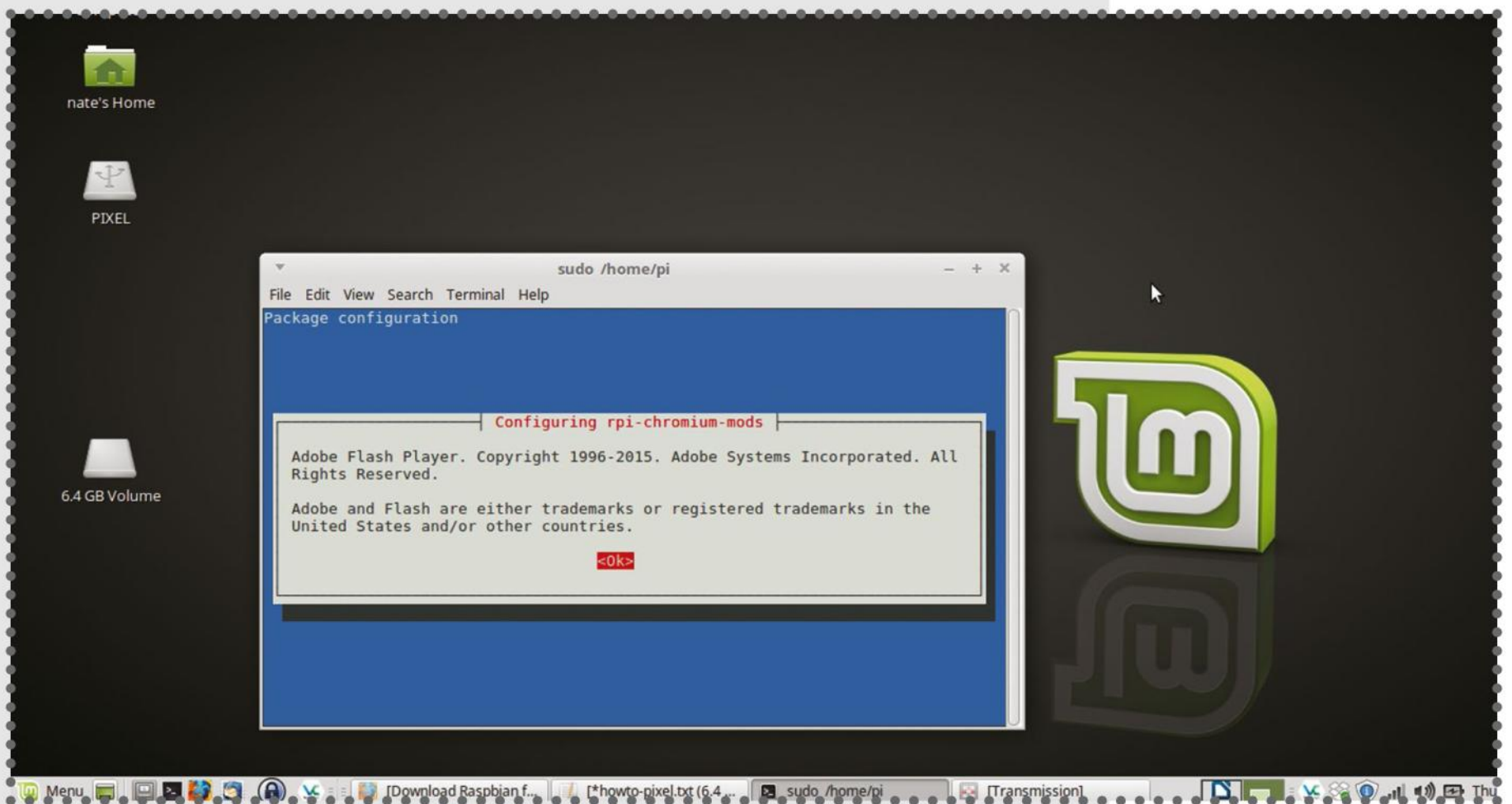
There are several ways to install Raspbian with Pixel. If you have Raspbian with NOOBS, you can restart your Pi by holding Shift and choose to reinstall. This will upgrade you to the latest version using Pixel but will also wipe your existing installation. Alternatively you can download the latest Raspbian Image from <https://www.raspberrypi.org/downloads/raspbian/> and follow the installation guide at <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>. If you use either of these methods, skip ahead to Step 5.





Above It's easy to upgrade Raspbian Jessie to Pixel using Terminal

```
sudo apt-get update
sudo apt-get dist-upgrade
```

03 Install Chromium

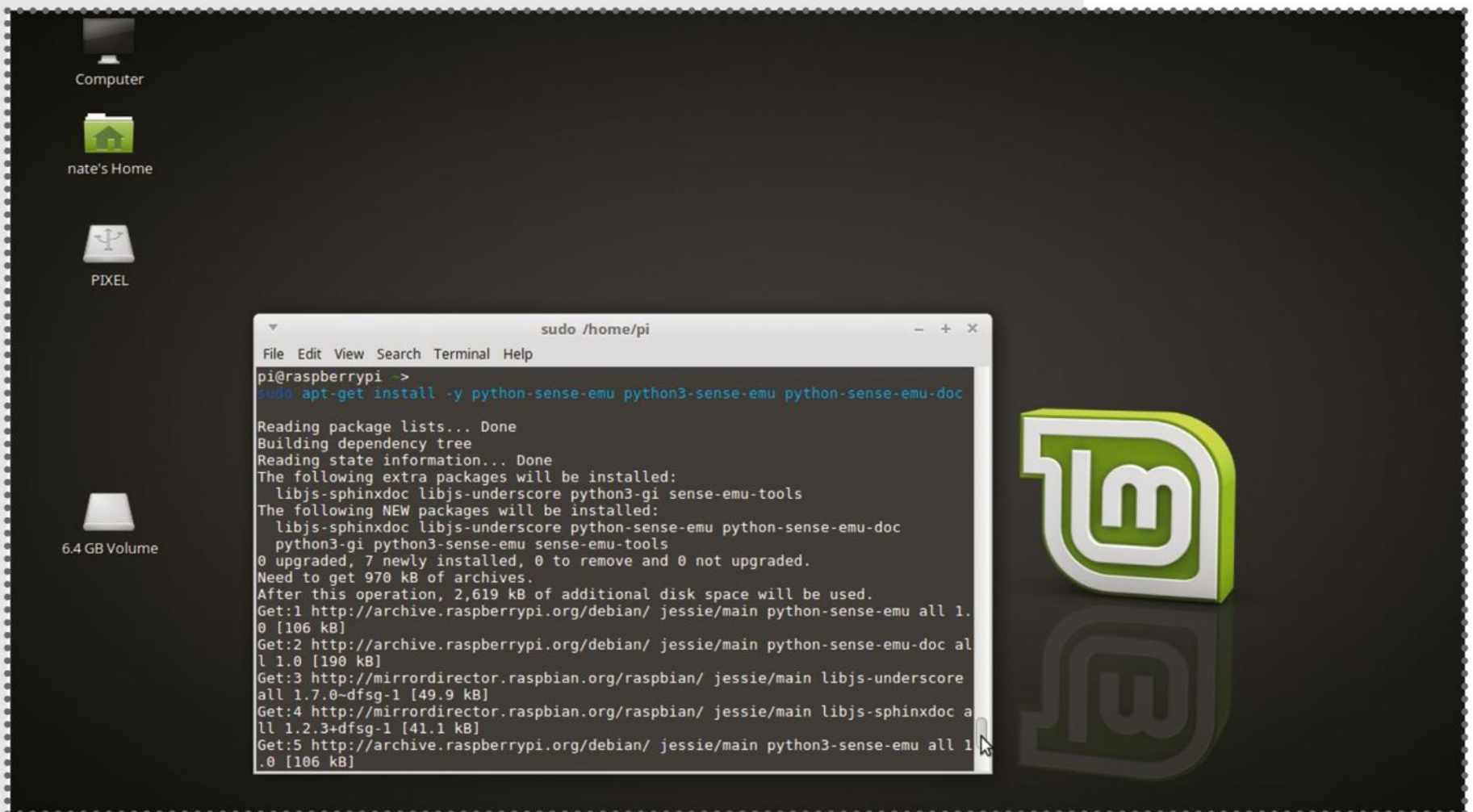
The latest version of Raspbian includes an initial release of Chromium. As we're performing a manual upgrade, you can install this with the command:

```
sudo apt-get install -y rpi-chromium-mods
```

Chromium comes bundled with the previously mentioned h264ify extension, which forces YouTube to use the Pi's hardware acceleration when streaming videos. The awesome adblocker uBlock Origin is also bundled to strip out resource-hungry adverts.

04 Install SenseHAT Emulator

This step is optional, but is included as the SenseHAT emulator is also included with the latest version of Raspbian. The SenseHAT is an add-on board for the Raspberry Pi with a range of sensors from a thermometer



to a gyroscope. The emulator allows developers to test code for devices without actually owning a SenseHAT itself. For more information see <http://www.raspberrypi.org/blog/desktop-sense-hat-emulator/>

Run this command to install all necessary files:

```
sudo apt-get install -y python-sense-emu  
python3-sense-emu python-sense-emu-doc
```

05 Load Pixel Desktop

Reboot the Pi to see your new, shiny Pixel Desktop. You may see a message stating that your previous configuration files have been overwritten. Click OK to dismiss this. At this stage you might also want to change the default wallpaper. Right click anywhere on the desktop and click Desktop Preferences. Click the Wallpaper menu to choose from any of the stunning options. Our current favourite is Mountain. If you're curious, the EXIF data in each image will tell you where it was taken.

06 Examine your interfaces

Pixel's Desktop now offers the option to disable both Bluetooth and Wi-Fi completely from within the desktop environment with the click of a button. Simply click on the relevant icon and turn it off. Head over to **Menu>Preferences>Raspberry Pi Configuration>Interfaces** and click **Enabled** under VNC. The VNC icon will appear at the top-right of the desktop. Click this to view your Pi's private IP address for VNC clients.

07 Import bookmarks into Chromium

You can open Chromium by clicking the web browser icon at the top-left of the screen. If you want to import your bookmarks from Epiphany, open Terminal on the Pi and run:

```
epiphany-browser.
```

Click the Settings gear icon on the top-right and choose **Edit Bookmarks**. On the window that opens you'll have the option to export your bookmarks to HTML format. Close Epiphany and choose the blue link **Import Bookmarks** in Chromium to add them there.

08 Change the Appearance Settings

While the default options make for a stunning desktop, you may wish to perform some small tweaks at this stage, particularly if this is a new install of Raspbian. In the main menu, head over to **Preferences>Appearance Settings**. If you're used to a menu bar at the bottom of the screen, change the **Position** setting to **Bottom**. You can also change the size of the bar to **medium** or **small**.

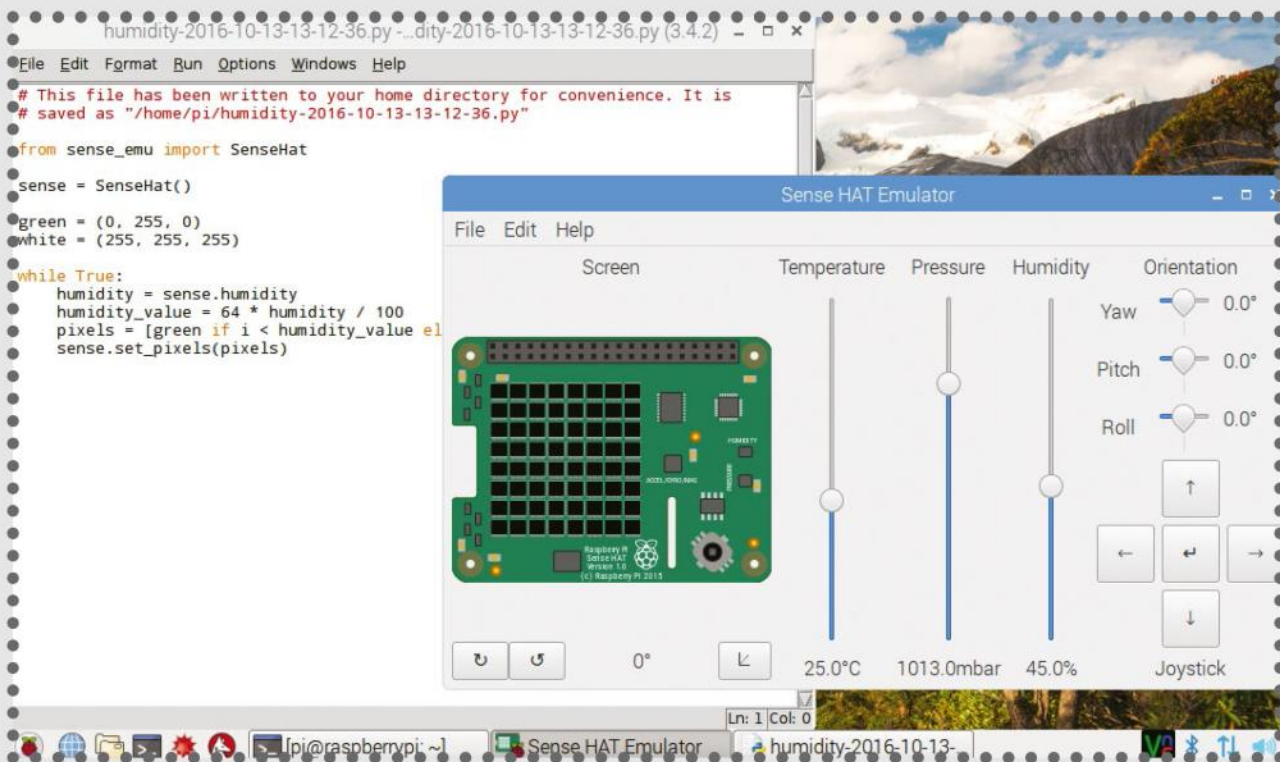
Invisible icons

Some of the newly designed icons aren't immediately visible as none of the default applications belong in the **Engineering** or **Education** categories. If you're curious about this, head over to `/usr/share/icons/PiX` to take a peek at the icons. Alternatively click on **Menu>Preferences>Add/Remove software** to see the categories and their respective sleek icons. It is possible to modify the icons for individual apps. Check Step 15 for more information.



FIND
MORE
FREE
MAGAZINES

FREEMAGS.CC



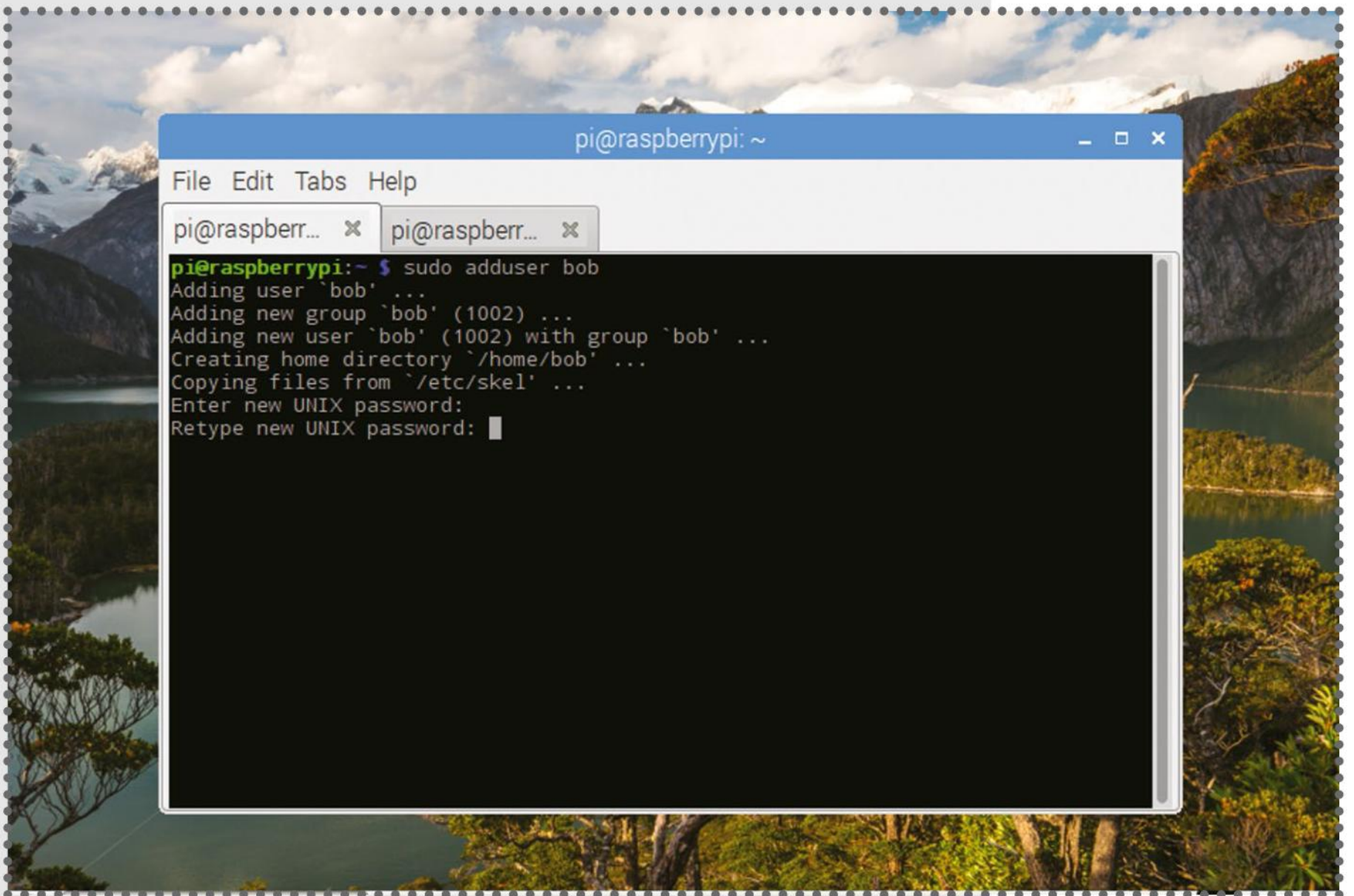
09 Test SenseHAT Emulator

If you installed the SenseHAT Emulator previously, you can launch it from the Programming menu in Applications. The Emulator comes bundled with around a dozen example scripts to get you started. Simply click the File menu at the top-left of the emulator window, then Open Example. There are beginner, intermediate and advanced projects. If your interest has been piqued, SenseHATs are currently available online from the Pi Hut for £30. See <https://thepihut.com/products/raspberry-pi-sense-hat-astro-pi>.

10 Remotely access your Pi

Now the Raspberry Pi has a rich desktop environment and modern web browser, you may wish to use it for a work or home computer. To make the Pi require a password on startup, open Terminal on or connect via SSH and run the command:

```
sudo nano /etc/lightdm/lightdm.conf
```



Scroll down to the **line autologin-user=pi** and put a hash (#) at the start. Press Ctrl+X, then Y, then return to save and exit. Remember the default password is 'raspberry'.

Above Pixel also allows you to add password protection and multiple users

11 Set up a VNC server

If you followed the previous step, you may want to add extra user accounts for your family or colleagues. First open Terminal on the Pi or connect via SSH and then run the following command:

```
passwd.
```

This will allow you to change the default password 'raspberry' to something more meaningful. Once you have done this add more users with the command:


```
sudo adduser  
(e.g sudo adduser bob)
```

Now enter the password for the new user twice when prompted to create the account.

12 Practise resizing windows

One blessing of the old Raspbian desktop is that window frames were quite thick and so it was easy to hover the mouse over them and resize. Pixel includes much slicker, thinner windows so it can be tricky at first to resize them. Fortunately in the latest version of Raspbian the grab handles now extend outside the window, so even if the mouse is just outside the frame, you'll see the cursor change.

13 Configure splash screen

Pixel's sedate splash screen on boot may not sit well with some people's principles; the previous text boot is also very useful for detecting errors. If you wish to go back to the old style boot screen, open Terminal or connect via SSH and run the command:

```
sudo nano /boot/cmdline.txt
```

Use the right arrow to find the text 'quiet splash' and delete it. Use Ctrl+X, then Y, then return to save and exit.

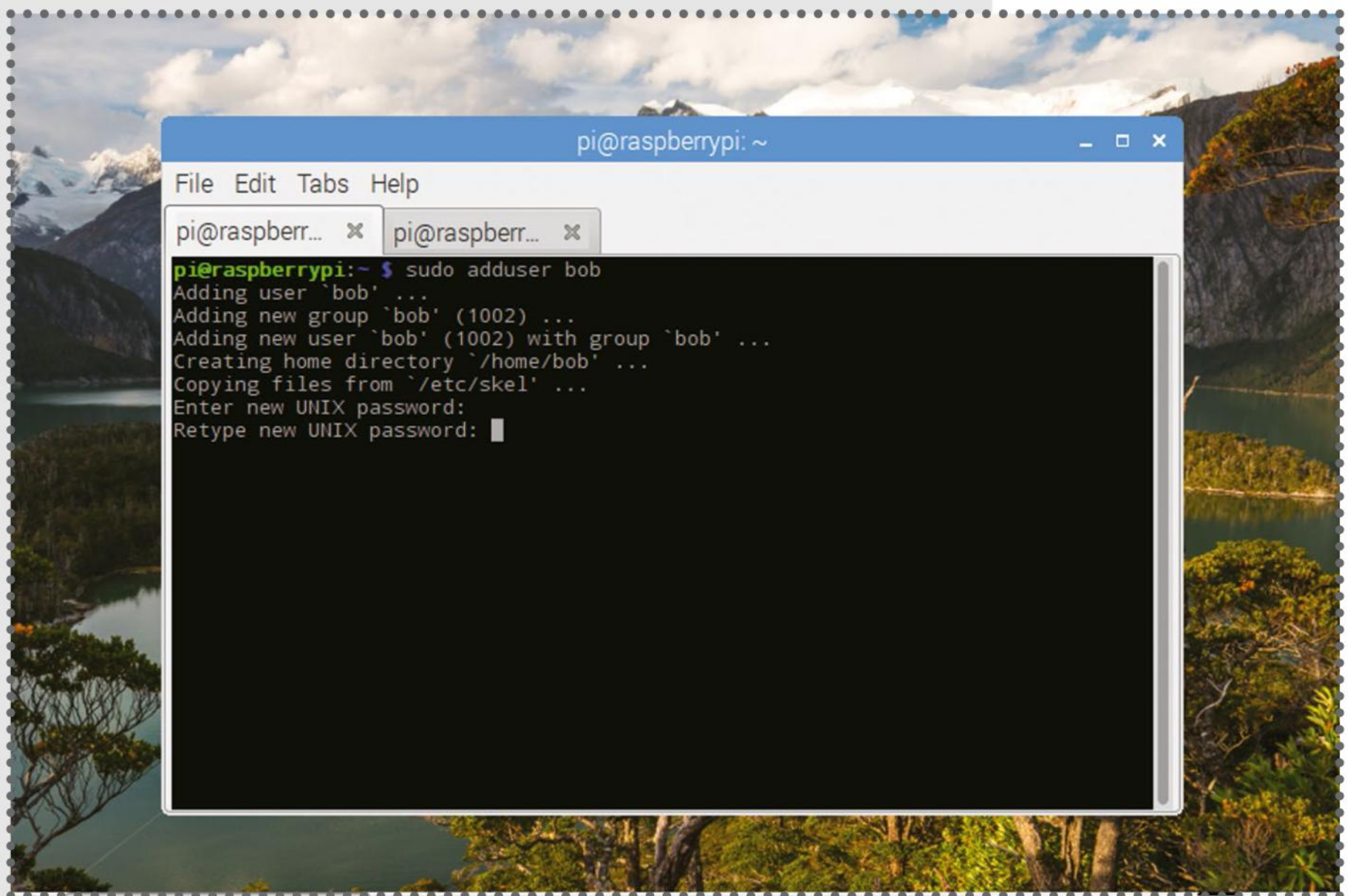
14 Add temperature/voltage monitors

In previous versions of Raspbian, if the Pi was overheating or underpowered, you may have noticed crude yellow and red squares appearing in the corner of

the screen. These have now been replaced with pictures of a thermometer and lightning bolt respectively. On the basis that prevention is better than cure, right-click the task bar and Add Panel Items to display the system temperature and voltage if you wish.

15 Explore your icons

The icons have been painstakingly redrawn for Pixel by Sam Alder and Alex Carter, who also illustrate the graphics of the official Raspberry Pi website. You can find all the system icons in **/usr/share/icons/PiX**. The icons are very professional and easy on the eye, however if you wish to replace any of them, make sure to try and use a .PNG image and to give it the exact same name as the image you're replacing, for example launch.png.

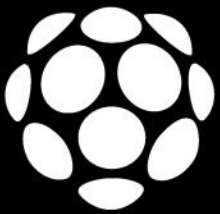




How to turn a Raspberry Pi into a walkie talkie

The talkiepi boasts an intuitive push-to-talk interface that's perfect for kids and big kids alike. Creator Daniel Chote talks us through this personal project





Where did the idea for talkiepi stem from?

I had the idea a few years ago to use a Raspberry Pi as a walkie talkie, but it wasn't until I watched Stranger Things that I got the itch to do something for my kids. Kids nowadays have smartphones and tablets, but they don't have any simple single-purpose electronic devices that do one thing well. I had recently purchased a 3D printer, the Monoprice Select Mini 3D to be exact, and had started tinkering with Fusion 360, so I thought it would be a great project that I actually had all the tools and hardware to see through to completion.

Did you encounter any significant problems when building your Wi-Fi walkie talkie?

It was about keeping it simple and kid-proof. I found the cheapest possible way to add a microphone and speaker to the Raspberry Pi, then came up with a very simple GPIO interface for status display and push-to-talk. I found an existing open source project that used Mumble, so I heavily modified that for my use. 3D modelling was pretty easy with Fusion 360.

Initially, I just wanted 'something', but then quickly realised that it needed to look cool, so I took some design cues from Eighties-style walkie talkies and integrated that into an enclosure that could cleanly house the Raspberry Pi and speakerphone PCB without it being too overcrowded. With the simplicity of the project, there wasn't really much to go wrong. I think the biggest problem is that this version of the talkiepi is a little too big for small hands. I plan to fix that by building a much smaller version using a Pi Zero.

You used a 3D printer for certain parts, is it essential to have access to a 3D printer for this project?

Not at all. You could assemble this in anything really. I 3D



Daniel Chote

Daniel Chote is a father of two currently living in Pennsylvania USA. He likes making things in code, by hand and with his trusted 3D printer.



printed an enclosure simply because I could. It also makes it a little more kid-proof.

How was the Raspberry Pi 3 implemented into the project, and how have you found working with it?

The Raspberry Pi 3 was chosen because of its overall performance and cost factor. The project needed to have Wi-Fi, and it seemed like the simplest route to get that. However, I had a lot of trouble getting the Raspberry Pi 3 Wi-Fi to work well enough on my home network, and opted for a USB dongle in the end anyway. One of my friends has a talkiepi running with the Raspberry Pi 3 Wi-Fi on his network without any issues, so it must just be an access point-specific problem. I have talkiepi running on a Pi Zero with a HubPiWi, and it seems to work quite well; it will occasionally drop off the network, but, for the most part, it works.

Mumble won't be a program that many of our readers will have used, what sort of role did it play in talkiepi?

I was initially thinking that I would use SIP, but was aware of Mumble and decided to see if someone had finally written a command-line client for it. Previously there was just the QT Mumble client, which is a little slow on the Raspberry Pi. I came across Gumble (<https://github.com/layeh/gumble>)



Raspberry Pi 3 Model B
microSD card

US Robotics USB
speakerphone

5 x M3 nylon screws + 1
M3x20 nylon standoff

GPIO header connector

2 x 5mm LEDs

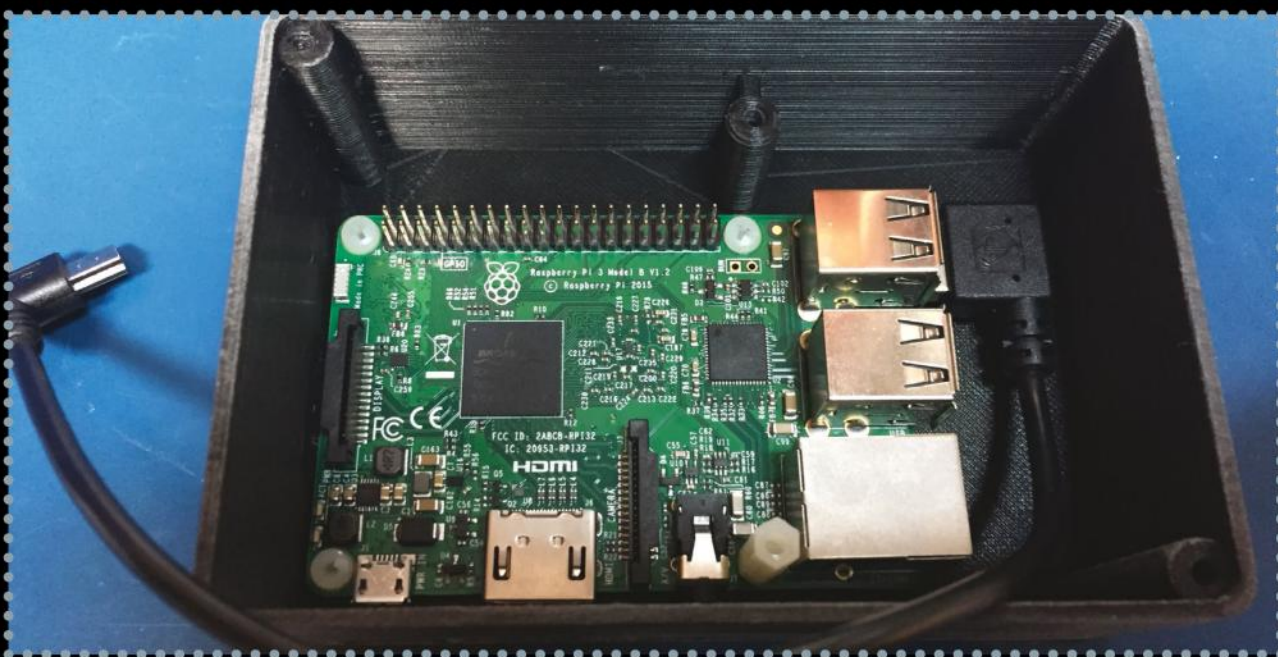
Pushbutton with LED

3 x 330ohm resistors

2 x M3x15 bolts

2 x M3x25 bolts

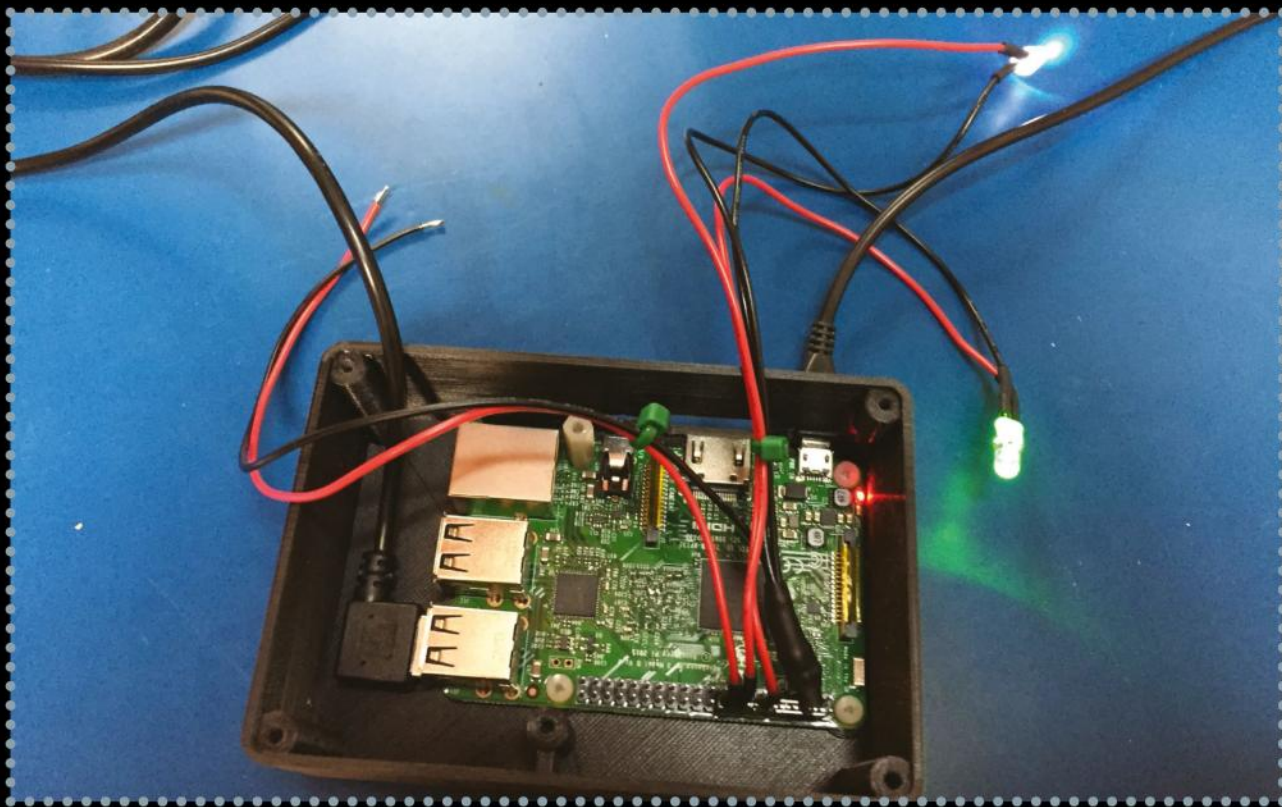
2 x M3x10 bolts and
nuts



and tested it on the Raspberry Pi 3. It worked right out of the gate, so Gumble became the foundation for talkiepi. Mumble is really nice because it is secure (client certificates) and supports server-side ACLs. Using ACLs you can restrict which channels any given user can interact with, letting you set up groups of talkiepis that can all use the same server, but have their own channel to interact in.

Do you feel you'll continue to develop talkiepi and is there anywhere else you feel it could be utilised?

I will definitely continue to develop and maintain it. The next few months for me are busy with work and family things, but I plan to release 3D models for the Pi Zero version, and an even smaller version that is just a Pi Zero, battery, and exposed USB port for a USB headset. I have an OLED display that I would like to integrate into the project to display Wi-Fi signal strength and general channel information; support for that will end up in the talkiepi codebase shortly. I could see this being used for a lot of practical things, such as restaurant staff communication. It can be built cheaply and is very simple to maintain once it's running.



Your website is full of amazing projects, have you got anything specific lined up in the near future?

Thanks! I will be finishing up the Planetarium Projector, that's my next 'must finish' project. I'm thinking that I will start getting into Raspberry Pi robotics next. I really like the idea of Formula Pi and, as my kids get older, I think they'll really get a kick out of it.

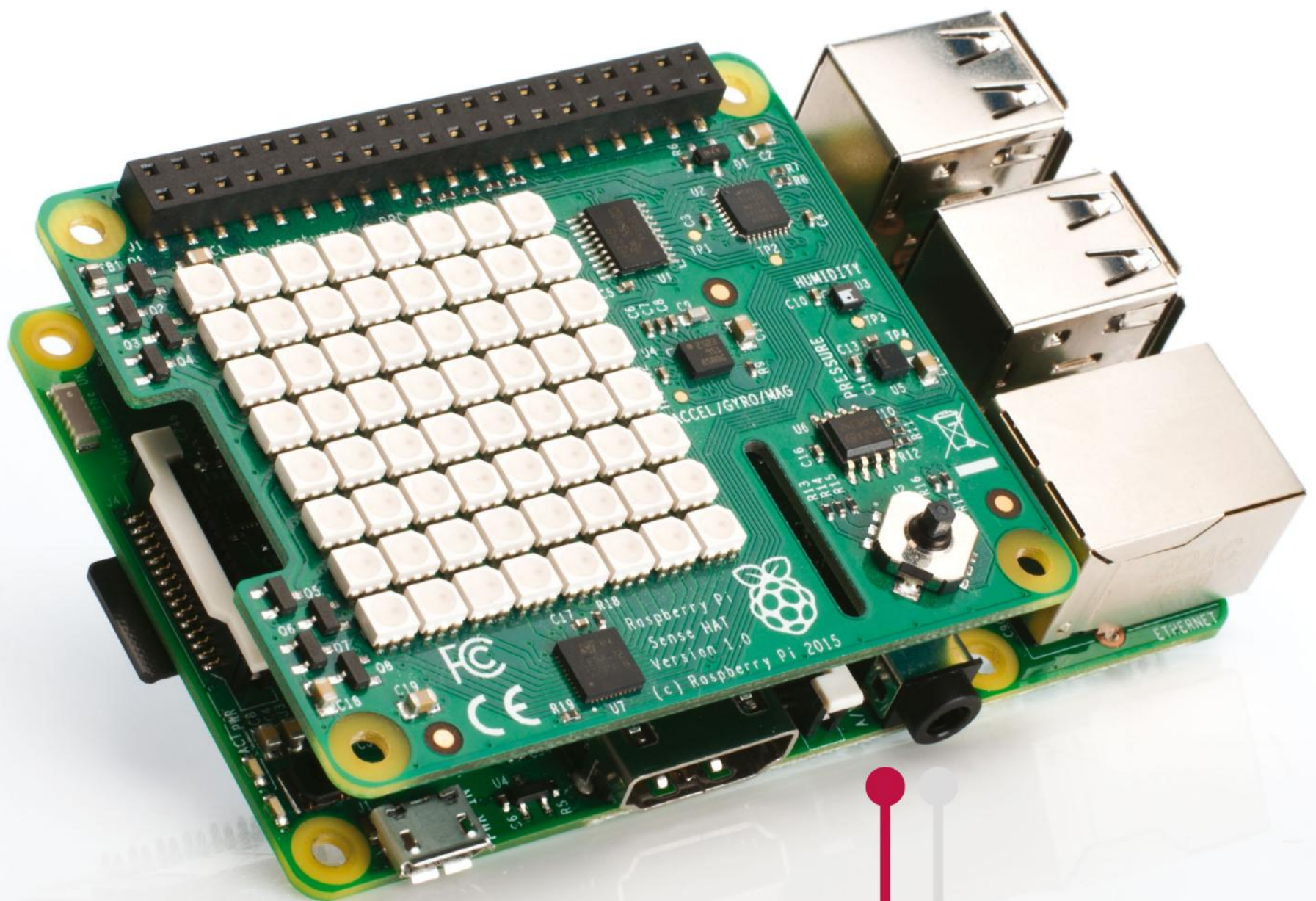
Keep up with Daniel Chote's latest Raspberry Pi projects on his blog, <http://projectable.me>.

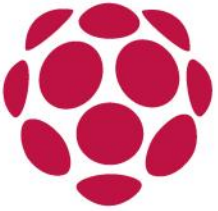




Make an egg-drop game with the Sense HAT

Use the same hardware that Major Tim Peake used on the ISS and code your own drop-and-catch game





Some of the most basic and repetitive games are the most fun to play. Consider Flappy Bird, noughts and crosses or even catch. This tutorial shows you how to create a simple drop-and-catch game that makes excellent use of some of the Sense HAT's features. Start off by coding an egg – a yellow LED – to drop each second, and a basket – a brown LED – on the bottom row of LEDs. Use the Sense HAT's accelerometer to read and relay back when you tilt your Sense HAT left or right, enabling you move the basket toward the egg. Successfully catch the egg and you play again, with a new egg being dropped from a random position... But, if you miss one, then it breaks and it's game over! Your program will keep you up to date with how you are progressing, and when the game ends, your final score is displayed. If you don't own a Sense HAT, you can use the emulator that is available on the Raspbian with PIXEL operating system. You can also see the Egg Drop game in action here:

<https://youtu.be/QmjHMzuWlqI>

01 Import the modules

First, open your Python editor and import the SenseHAT module, line 1. Then import the time module, line 2, so you can add pauses to the program. The random module, line 3, is used to select a random location from the top of the LEDs, from which the egg will drop. To save time typing 'SenseHAT' repeatedly, add it to a variable, line 4. Finally, set all the LEDs to off to remove the previous score and game data, line 5.

```
from sense_hat import SenseHat
```



line 3 to False, this means the game is not over. The position of each LED on the matrix is referred to by the co-ordinates x and y, with the top line being number 0 down to number 7 at the bottom. Create a variable to hold the position of the basket, which is set on the bottom line of the LEDs, number 7. Finally, set the score to zero.

```
global game_over
global score
game_over = False
basket_x = 7
score = 0
```

03 Measure the basket movement: part 1

The basket is controlled by tilting your Sense HAT to the left or right, which alters the pitch. Create a function to hold the code, which will be used to respond to the movement and move the basket. On line 1, name the function; include the pitch reading and the position of the basket, basket_x. Use sense.set_pixel to turn on one LED at the bottom-right of the LEDs matrix, the co-ordinates (7,7), line 2. Then set the next position of the basket to the current position so that the function is updated when it runs again. This updates the variable with the new position of the basket and turns on the corresponding LED. This has the effect of looking like the basket has moved.

```
def basket_move(pitch, basket_x):
    sense.set_pixel(basket_x, 7, [0, 0,
0])
    new_x = basket_x
```



Johan Vinet has some excellent and inspirational examples of 8×8 pixel art, which include some famous characters and will show you what you can create with 64 pixels of colour.

04 Measure the basket movement: part 2

The second part of the function consists of a conditional which checks the pitch and the basket's current position. If the pitch is between a value of 1-179 and the basket is not at position zero, then the Sense HAT is tilted to the right and therefore the basket is moving to the right. The second condition checks that the value is between 359 and 179, which means that the tilt is to the left, line 3. The last line of code returns the x position of the basket so it can be used later in the code – see Step 13.

```
if 1 < pitch < 179 and basket_x != 0:
    new_x -= 1
elif 359 > pitch > 179 and basket_x != 0:
    new_x += 1
return new_x,
```


05 Create images for your game

Images are built up of pixels that combine to create an overall picture. Each LED on the matrix can be automatically set from an image file. For example, an image of a chicken can be loaded, the colours and positions calculated, and then the corresponding LEDs enabled. The image needs to be 8×8 pixels in size so that it fits the LED matrix. Download the test picture file, **chicken.png**, and save it into the same folder as your program. Use the code here in a new Python window to open and load the image of the chicken (line 3). The Sense HAT will do the rest of the hard work for you.

```
from sense_hat import SenseHat
sense = SenseHat()
sense.load_image("chicken.png")
```

06 Install Tweepy

The simplest method to create your own image with the LEDs is a superb on-screen program that enables you to manipulate the LEDs in real-time. You can change the colours, rotate them and then export the image as code or as an 8×8 PNG file. First, you need to install Python PNG library; open the Terminal window and type:

```
sudo pip3 install pypng
```

After this has finished, type:

```
git clone https://github.com/jrobinson-uk/RPi_8x8GridDraw
```



Once the installation has completed, move to the RPi folder:

```
cd RPi_8x8GridDraw
```

Now enter the command:

```
python3 sense_grid.py
```

...to run the application.

07 Create and export your image

The Grid Editor enables you to select from a range of colours displayed down the right-hand side of the window. Simply choose the colour and then click the location of the LED on the grid; select 'Play on LEDs' to display the colour on the Sense HAT LED. Clear the LEDs using the Clear Grid button and then start over. Finally, when exporting the image, you can either save as a PNG file and then apply the code in the previous step to display the picture, or you can export the layout as code and import that into your program.

08 Display a message: the game begins

Now you have an image, you are ready to create the function that controls the whole game. Create a new function, line 1, called main, and add the code:

```
sense.show_message
```

...to display a welcome message to the game, line 3. The values 255, 255 and 0 refer to the colour of the message (in this example, yellow). Edit these to



choose your own preferred colour.

```
def main():  
    global game_over  
    sense.show_message("Egg Drop", text_  
colour = [255, 255, 0])
```

09 Display your start image

Once the start message has scrolled across the Sense HAT LED matrix, you can display your game image, in this example a chicken. Due to the orientation of the Sense HAT and the location of the wires, you'll need to rotate it through 90 degrees so it faces the player, line 1. Load the image with the code `sense.load_image`, line 2. Display the image for a few seconds using `time.sleep()`, line 3. Note that the lines from now on are indented in line with the previous line.

```
sense.set_rotation(90)  
sense.load_image("chick.png")  
time.sleep(2)  
sense.set_rotation()
```

10 Count down to the game starting

Once the start image has been displayed, prepare the player to get ready for the game with a simple countdown from three to one. First create a list called `countdown`, which stores the values 3, 2, and 1, line 1. Use a for loop, line 2, to iterate through each of the numbers and display them. This uses the code `sense.show_message(str(i))` to display each number on the LEDs. You can adjust the colour of the number



using the three-part RGB values, text_colour = [255, 255, 255]), line 3.

```
countdown = [3, 2, 1]
for i in countdown:
    sense.show_message(str(i), text_
colour = [255, 255, 255])
```

11 Set the egg and basket

As the game starts, set the horizontal position, the 'x' position, of the basket to 7; this places the basket in the bottom right-hand corner of the LED matrix. Now set the x position of the egg at a random position between 0 and 7, line 2. This is at the top of the LED matrix and ensures that the egg does not always fall from the same starting point. Last, set the egg's y value to 0 to ensure that the egg falls from the very top of the LED matrix, line 3.

```
basket_x = 7
egg_x = random.randrange(0,7)
egg_y = 0
```

12 Display the egg and basket

In the previous step, you set the positions for the egg and the basket. Now use these variables to display them. On line 1, set the egg using the code sense.set.pixel followed by its x and y co-ordinates. The x position is a random position between 0 and 7, and the y is set to 0 to ensure that the egg starts from the top. Next, set the colour to yellow. (Unless your egg is rotten, in which case set it to green (0,255, 0). Next, set the basket position using the same code, line



2, where the x position is set to 7 to ensure that the basket is displayed in the bottom right-hand LED. Set the colour to brown using the values 139, 69, 19.

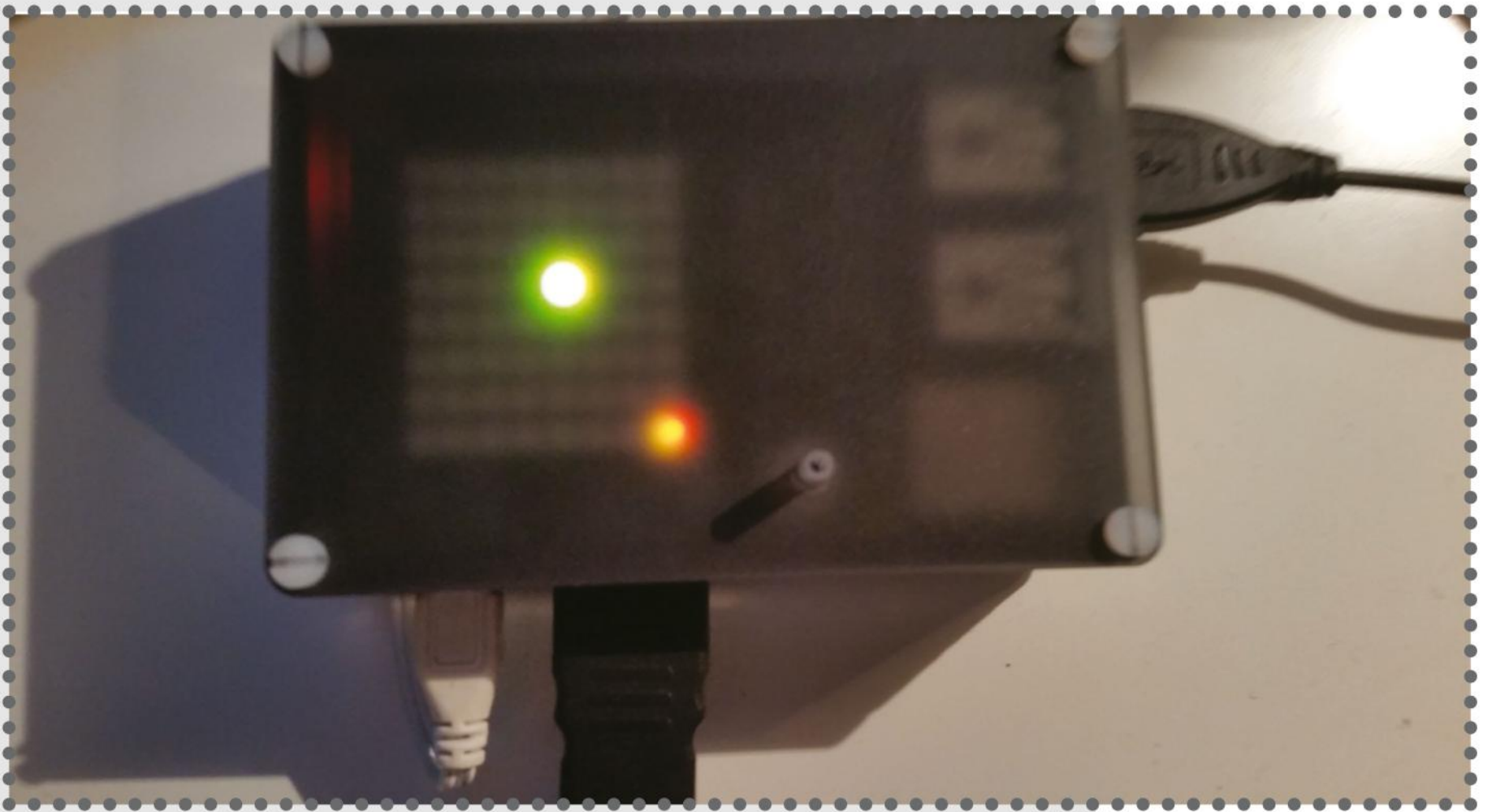
```
sense.set_pixel(egg_x, egg_y, [255, 255, 0])
sense.set_pixel(basket_x, 7, [139, 69, 19])
time.sleep(1)
```

13 Move the basket: part 1

Begin by checking that the game is still in play (the egg is still dropping), checking that the `game_over` variable is `False`, line 1. On line 2, import the score. Next, take a reading of the 'pitch' of the Sense HAT, using the code `sense.get_orientation()['pitch']`, line 3. Note that this is the value derived from the function you created in steps 4 and 5. The final line of code uses the function to turn off the LED that represents the basket and then looks at the value of the pitch, determining if the Sense HAT is tilted to the left or right, and then either adds or subtracts one from the x position of the current LED. This has the effect of selecting either the adjacent left or right LED to the current LED. Finally, update the `basket_x` value with the new position value.

```
while game_over == False:
    global score
    pitch = sense.get_orientation()
    ['pitch']
    basket_x, = basket_move(pitch,
basket_x)
```





14 Move the basket: part 2

Your program has now calculated the new position of the basket. Next, turn on the relevant LED and display the basket in its new position. On line 1, use the code `sense.set_pixel(basket_x, 7, [139, 69, 19])` to set and turn on the LED; `basket_x` is the value calculated in the previous step using the function in steps 4 and 5. Add a short time delay to avoid over-reading the pitch, line 2. You now have a basket that you can move left and right.

```
sense.set_pixel(basket_x, 7, [139, 69, 19])
time.sleep(0.2)
```

15 Install Tweepy

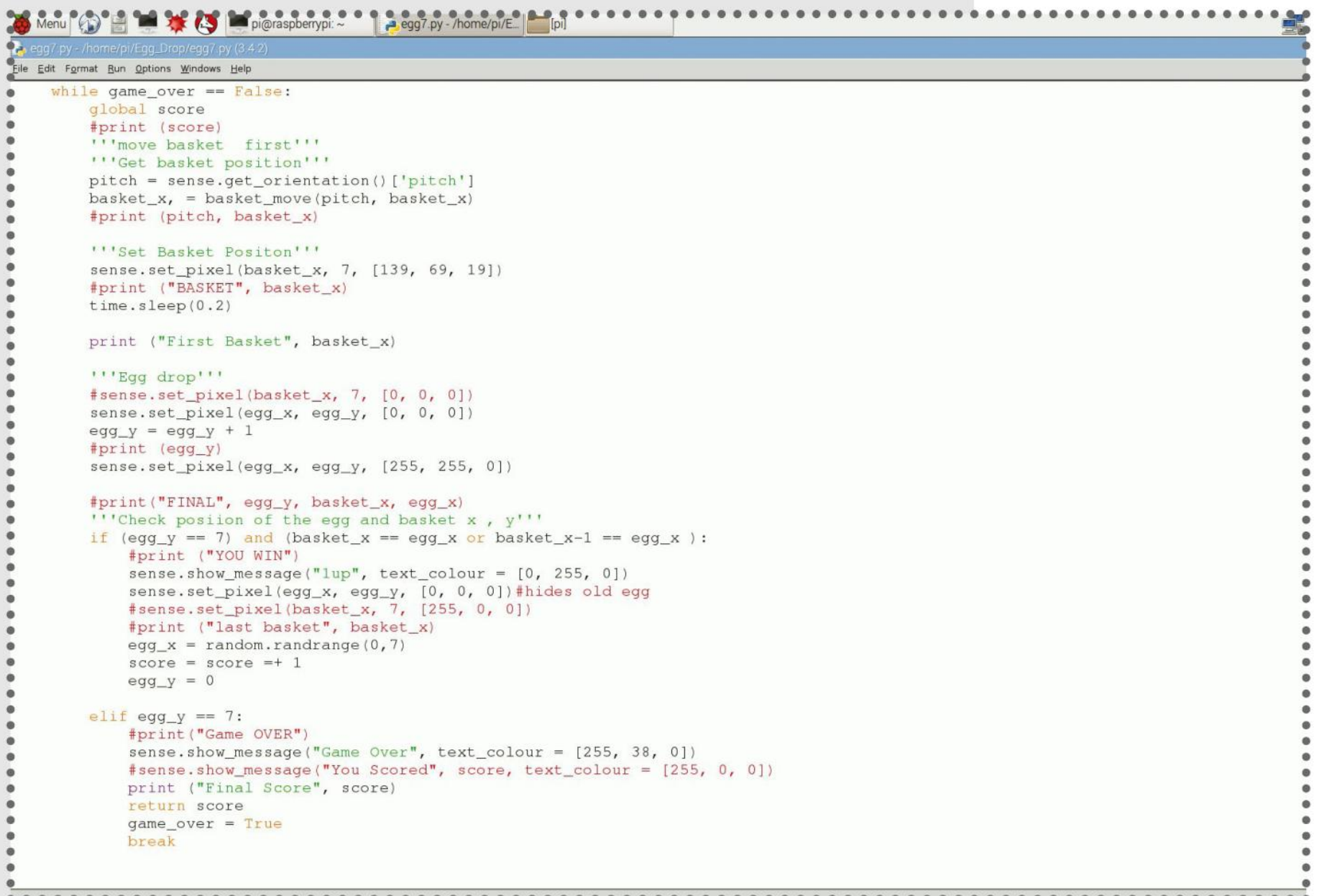
The egg is dropped from a random position from one of the LEDs across the top line. To make it appear to

be dropping, first turn off the LED that represents the egg using the code `sense.set_pixel(egg_x, egg_y, [0, 0, 0])`. The values 0, 0, 0, refer to black and therefore no colour will be displayed; it will appear that the egg is no longer on the top line.

```
sense.set_pixel(egg_x, egg_y, [0, 0, 0])
```

16 Drop the egg: part 2

Since the egg drops downwards, you only need to update the y axis position. Do this on line 1 by updating the `egg_y` variable using the code `egg_y = egg_y + 1`, which means it will change from an initial value of zero to a new value of one. (The next time the 'game loop' runs, it will update to two and so on until the egg reaches the bottom of the matrix, a value of



```
Menu [pi@raspberrypi: ~] egg7.py - /home/pi/E [pi]
egg7.py - /home/pi/Egg_Drop/egg7.py (3.4.2)
File Edit Format Run Options Windows Help

while game_over == False:
    global score
    #print (score)
    '''move basket first'''
    '''Get basket position'''
    pitch = sense.get_orientation()['pitch']
    basket_x, = basket_move(pitch, basket_x)
    #print (pitch, basket_x)

    '''Set Basket Positon'''
    sense.set_pixel(basket_x, 7, [139, 69, 19])
    #print ("BASKET", basket_x)
    time.sleep(0.2)

    print ("First Basket", basket_x)

    '''Egg drop'''
    #sense.set_pixel(basket_x, 7, [0, 0, 0])
    sense.set_pixel(egg_x, egg_y, [0, 0, 0])
    egg_y = egg_y + 1
    #print (egg_y)
    sense.set_pixel(egg_x, egg_y, [255, 255, 0])

    #print("FINAL", egg_y, basket_x, egg_x)
    '''Check position of the egg and basket x , y'''
    if (egg_y == 7) and (basket_x == egg_x or basket_x-1 == egg_x ):
        #print ("YOU WIN")
        sense.show_message("lup", text_colour = [0, 255, 0])
        sense.set_pixel(egg_x, egg_y, [0, 0, 0])#hides old egg
        #sense.set_pixel(basket_x, 7, [255, 0, 0])
        #print ("last basket", basket_x)
        egg_x = random.randrange(0,7)
        score = score + 1
        egg_y = 0

    elif egg_y == 7:
        #print("Game OVER")
        sense.show_message("Game Over", text_colour = [255, 38, 0])
        #sense.show_message("You Scored", score, text_colour = [255, 0, 0])
        print ("Final Score", score)
        return score
        game_over = True
        break
```

seven). Once the y position is updated, display the egg in its new position, using `sense.set_pixel`, line 2. The egg will appear to have dropped down one LED toward the bottom.

```
egg_y = egg_y + 1
sense.set_pixel(egg_x, egg_y, [255, 255,
0])
```

17 Did you catch the egg?

At this stage in the tutorial, you have a falling egg and a basket that you can move left and right as you tilt the Sense HAT. The purpose of the game is to catch the egg in the basket, so create a line of code to check that this has happened, line 1. This checks that the egg is at the bottom of the LED matrix, ie in position 7, and that the basket's x position is the same value as the egg's x position. This means that the egg and the basket are both located in the same place and therefore you have caught the egg.

```
if (egg_y == 7) and (basket_x == egg_x
or basket_x-1 == egg_x ):
```

18 Success!

If you catch the egg in the basket, then you gain one point. Notify the player by scrolling a message across the LEDs using the `sense.show_message()`, line 1. Since you have caught the egg, it should disappear as it is in the basket; to do this set the 'egg' pixel to a colour value of 0, 0, 0. This turns off the egg LED, making the egg disappear. Note: these lines are indented.


```
sense.show_message("1up", text_
colour = [0, 255, 0])
sense.set_pixel(egg_x, egg_y, [0, 0,
0])
```

19 Set up for the next round

Since you caught the egg, you get to play the game again. Set a new egg to drop from a random x position on the LED matrix, line 1. Update your score by one point and then set the egg's y position to 0, line 3. This ensures that the egg is back at the very top of the LED matrix before it starts dropping.

20 What happens if you miss the egg?

Since the game is over, change the `game_over` variable to `True`, which will stop the game loop from running again and then runs the last line of the program.

```
game_over = True
break
```

21 Stop the game

If you catch the egg in the basket, then you gain one point. Notify the player by scrolling a message across the LEDs using the `sense.show.message()`, line 1. Write your own message and select a colour. Since you have caught the egg, it should disappear as it is in the basket; to do this set the 'egg' pixel to a colour value of 0, 0, 0. This basically turns off the egg LED, making the egg disappear. Note: these lines are indented.

```
sense.show_message("1up", text_
colour = [0, 255, 0])
sense.set_pixel(egg_x, egg_y, [0, 0,
0])
```

22 Start the game

The main instructions and game mechanics are stored in one function called `main()`, which holds most of the game structure and processes. Functions are located at the start of a program to ensure that they are loaded first, ready for the program to use. To start the game, simply call the function (line 1), add a small delay (line 2), and ensure all the LEDs are set to off before the game starts (line 3).

23 Display your final score

If you did not catch the egg, then the game is over and your score is scrolled across the LEDs. This uses the line `sense.show_message` and then pulls the value from the `global_score` variable; convert this value into a string using `str`, line 1. Your program is now completed; save the file and then run it. Press F5 on the keyboard to do this. After the opening image and message are displayed, the countdown will begin and the game will start. Can you catch the egg?

```
sense.show_message("You Scored " +
str(score),
text_colour = [128, 45, 255], scroll_speed
= 0.08)
```


The Code

OPTIMISATION

```
from sense_hat import SenseHat
###Egg Drop###
###Coded by dan_albred###

import time
import random
sense = SenseHat()
sense.clear()

global game_over
global score

game_over = False
basket_x = 7
score = 0

'''main pitch measurement'''
def basket_move(pitch, basket_x):
    sense.set_pixel(basket_x, 7, [0, 0, 0])
    new_x = basket_x
    if 1 < pitch < 179 and basket_x != 0:
        new_x -= 1
    elif 359 > pitch > 179 and basket_x != 7:
        new_x += 1
    return new_x,

'''Main game setup'''
def main():
    global game_over
```

```

'''Introduction'''
sense.show_message("Egg Drop", text_
colour = [255, 255, 0])
sense.set_rotation(90)
sense.load_image("chick.png")
time.sleep(2)
sense.set_rotation()

'''countdown'''
countdown = [3, 2, 1]
for i in countdown:
    sense.show_message(str(i), text_
colour = [255, 255, 255])

basket_x = 7
egg_x = random.randrange(0,7)
egg_y = 0
sense.set_pixel(egg_x, egg_y, [255,
255, 0])
sense.set_pixel(basket_x, 7, [139, 69,
19])
time.sleep(1)

while game_over == False:
    global score
    '''move basket first'''
    '''Get basket position'''
    pitch = sense.get_orientation()
['pitch']
    basket_x, = basket_move(pitch,
basket_x)

```



```

        '''Set Basket Positon'''
        sense.set_pixel(basket_x, 7, [139,
69, 19])
        time.sleep(0.2)

        '''Egg drop'''
        sense.set_pixel(basket_x, 7, [0, 0,
0])
        sense.set_pixel(egg_x, egg_y, [0,
0, 0])
        egg_y = egg_y + 1
        #print (egg_y)
        sense.set_pixel(egg_x, egg_y, [255,
255, 0])

        '''Check posiion of the egg and
basket x , y'''
        if (egg_y == 7) and (basket_x ==
egg_x or basket_x-1 == egg_x ):
            sense.show_message("1up", text_
colour = [0, 255, 0])
            sense.set_pixel(egg_x, egg_y,
[0, 0, 0])#hides old egg
            egg_x = random.randrange(0,7)
            score = score += 1
            egg_y = 0

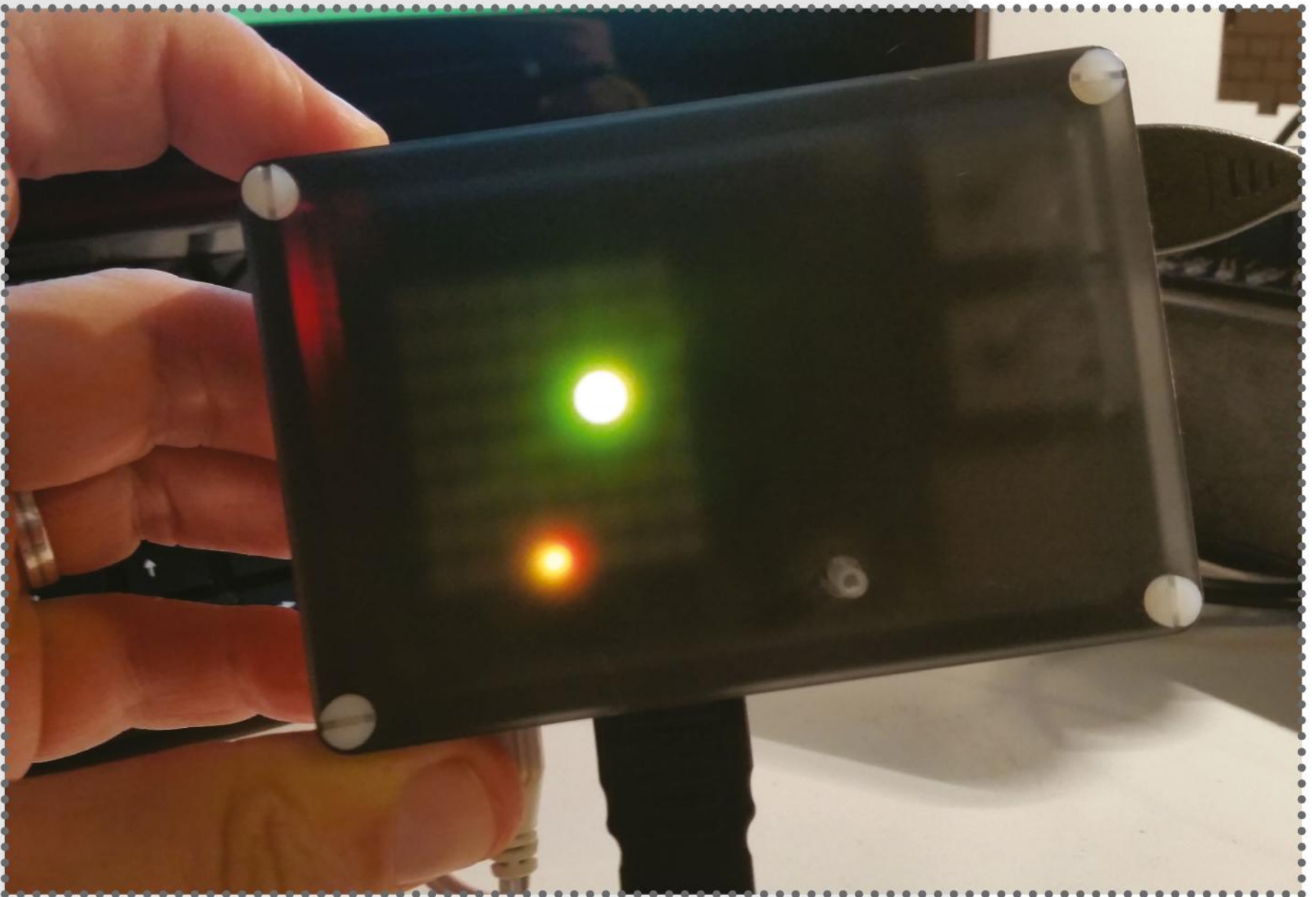
        elif egg_y == 7:
            sense.show_message("Game Over",
text_colour = [255, 38, 0])
            return score

```



```
game_over = True  
break
```

```
main()  
time.sleep(1)  
sense.clear()  
sense.show_message("You Scored " +  
str(score), text_colour = [128, 45, 255],  
scroll_speed = 0.08)
```





Hack a toy with the Raspberry Pi : Part 1

Learn how to master four simple hacks and embed them into a toy and bring it to life



Combining an old toy and a Raspberry Pi, you can embed a selection of components to create your own augmented toy that responds to user input. For example, take a £3 R2-D2 sweet dispenser, light it up, play music and stream a live web feed to your mobile device.

Part one of this two-part tutorial covers setting up four basic features: an LED for the eye, a haptic motor to simulate an electric shock, a webcam stream from a hidden camera and the Star Wars theme tune broadcast to a radio. You may choose to combine these with your own hacks or use them as standalone features in other projects. Don't feel limited to just using a Star Wars toy either – we used the R2D2 figure because it was cheap, available and popular, but there's no limit to the toys that you can experiment with. Action figures (provided that they can be disassembled and/or have a stand or cavity that can be used to hold the electronics) and plushy or cuddly toys both lend themselves well to this kind of maker project, especially if they accompany a movie or TV show that has recognisable music or sound effects



THE PROJECT ESSENTIALS

An old/new toy

Resistors

Small disc motor

LED

A radio

Small webcam

Female-to-female jumper
jerky wire



that you can make them broadcast. Part two next issue covers how to set up the toy and set up triggers to start features.

01 Set up an LED

LEDs are really easy to set up and control. They make a nice addition to your toy and can be used as eyes, flashing buttons or, in this example, R2-D2's radar eye. Take your LED and hold the longest leg; this is the positive arm and connects to the input. Wrap the resistor around the leg and attach to a female jumper jerky wire. Now take the negative arm and attach this to another jumper wire.

02 Attach the LED

Take the positive wire, the one with the resistor, to GPIO 17, which is physical pin number 11 (look at the far-left top pin and count down six pins). This pin will provide the current that powers the LED. Connect the other wire to any of the ground pins, (GND) 6, 9, 14, 20, 39; you may need to move this around later as you add more wires for the other components.

03 Light up the LED

To turn the LED on and off, use the `gpiozero` library, which was developed to simplify the interaction between code and a physical computer. Open the LX Terminal and type

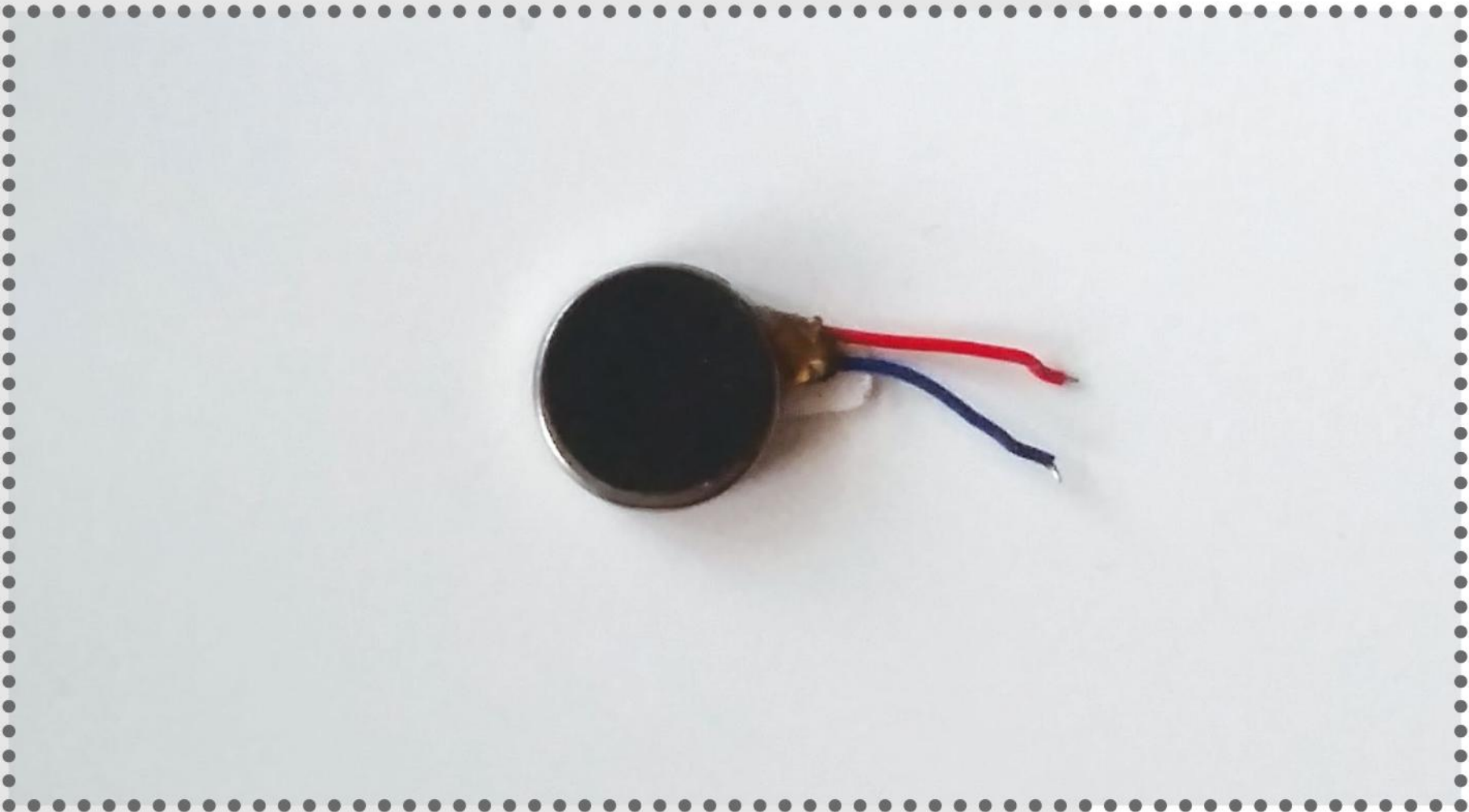
```
sudo apt-get install python3-gpiozero
```

to install the library. Once installed, open a new Python window and import the LED module (line 2 of the following code). Assign the pin number of the LED (line 3) and finally turn it on and off (lines 6 and 8). Save your code and run it to make the LED flash.

```
import time
from gpiozero import LED
led = LED(17)
led.off()
```

```
while True:
    led.on()
    time.sleep(0.5)
    led.off()
    time.sleep(0.5)
```





04 Add a vibration

You may want the toy to vibrate or shake when it is touched. R2-D2 is known for giving out electric shocks and a safe way to emulate this is to add haptic feedback, similar to that when you press a key on the screen of your mobile. Pimoroni stocks a suitable disc motor (bit.ly/29hIEla), which will deliver a short vibration. Take each of the wires and connect them each to a female-to-female jumper wire.

05 Wire up the disc motor

Take the positive wire from the motor (usually coloured red) and attach it to GPIO pin 9; this is physical pin number 21. The black wire connects to a ground pin – the nearest is physical pin 25; from pin 21, drop down two pins on the left and this is number 25. Start a new program or add the code to your existing program.

Import the RPi.GPIO library (line 1) and set the board to the BCM setting. This ensures that the GPIO numbering system is used.

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
import time
```

06 Turn the motor on

To enable the motor, first set the output pin, number 9. This tells the program that GPIO pin 9 is an output. Next, set the GPIO pin to 'HIGH' (line 2), this is the same as turning it on. Current flows through and your motor will turn on. Add a short pause (line 3) before setting the output to LOW, which turns the motor off. Play around with the timings to find the perfect pause for your needs.

```
GPIO.setup(9, GPIO.OUT)
GPIO.output(9, GPIO.HIGH)
time.sleep(5)
GPIO.output(9, GPIO.LOW)
```

07 Hack a web camera

A small web camera can be hidden within the toy as an eye or more discreetly within the body of the toy. This can be used to take photos or stream a live feed to a laptop or mobile device. Take your webcam and carefully strip away the plastic shell so you are left with the board and lens. Depending on the size of your toy, adjust the casing so that it can be hidden.

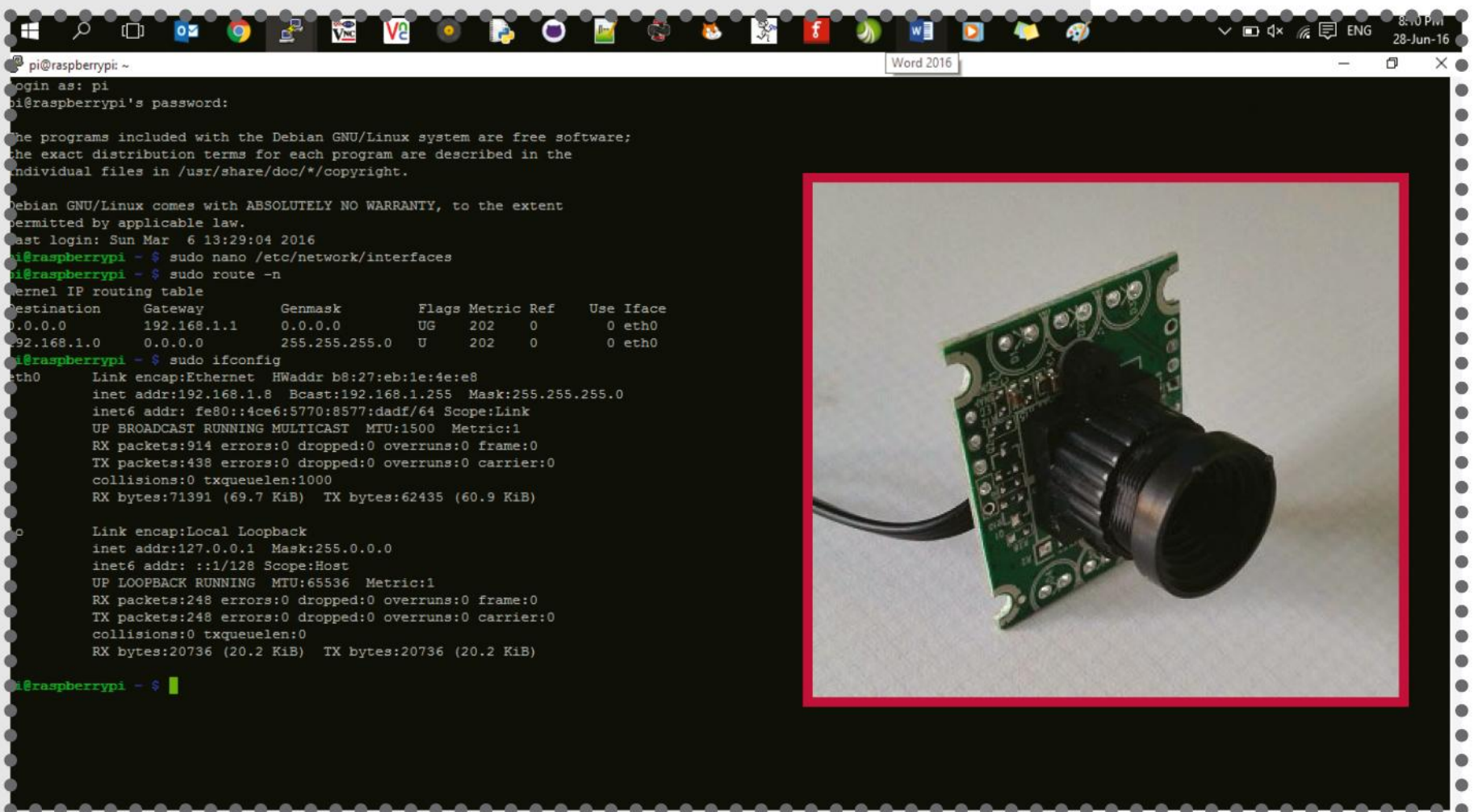
08 Set up a static IP address

R2-D2 in action

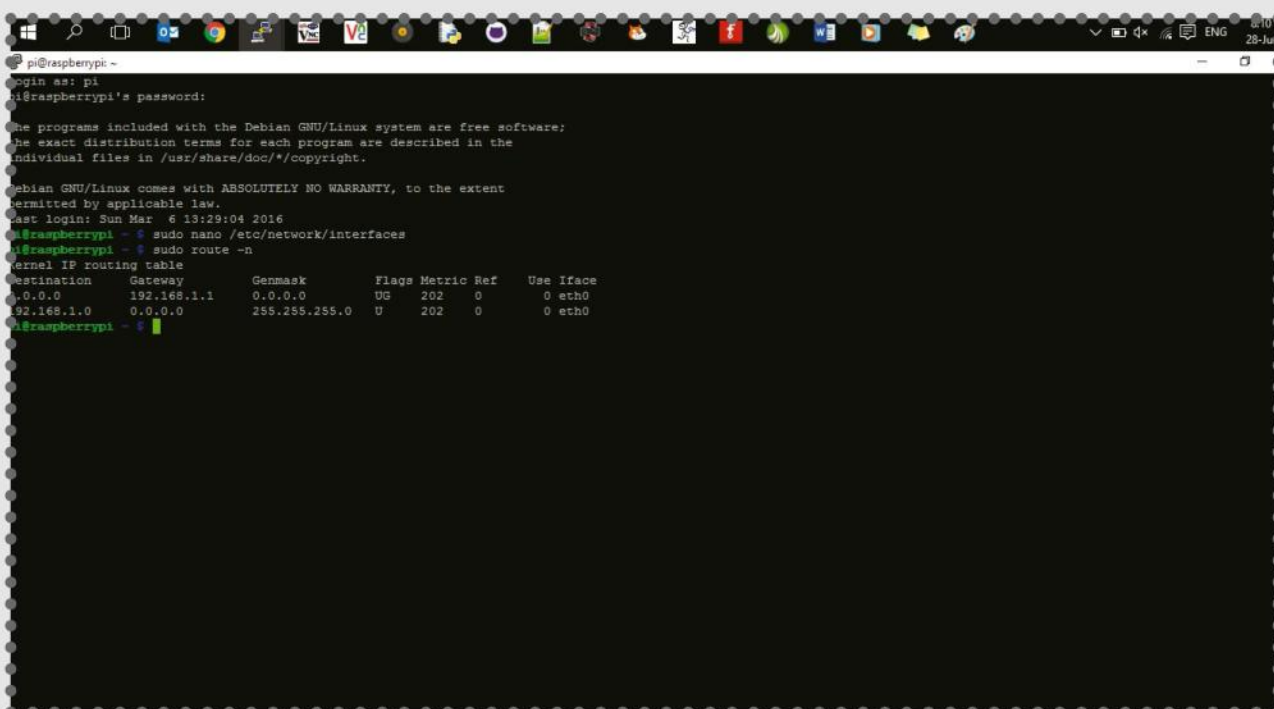
Check out the video of the completed R2-D2 toy hack and see the features in action. This may give you some ideas for your own toy hack.

youtu.be/VnOsUaS5jSY





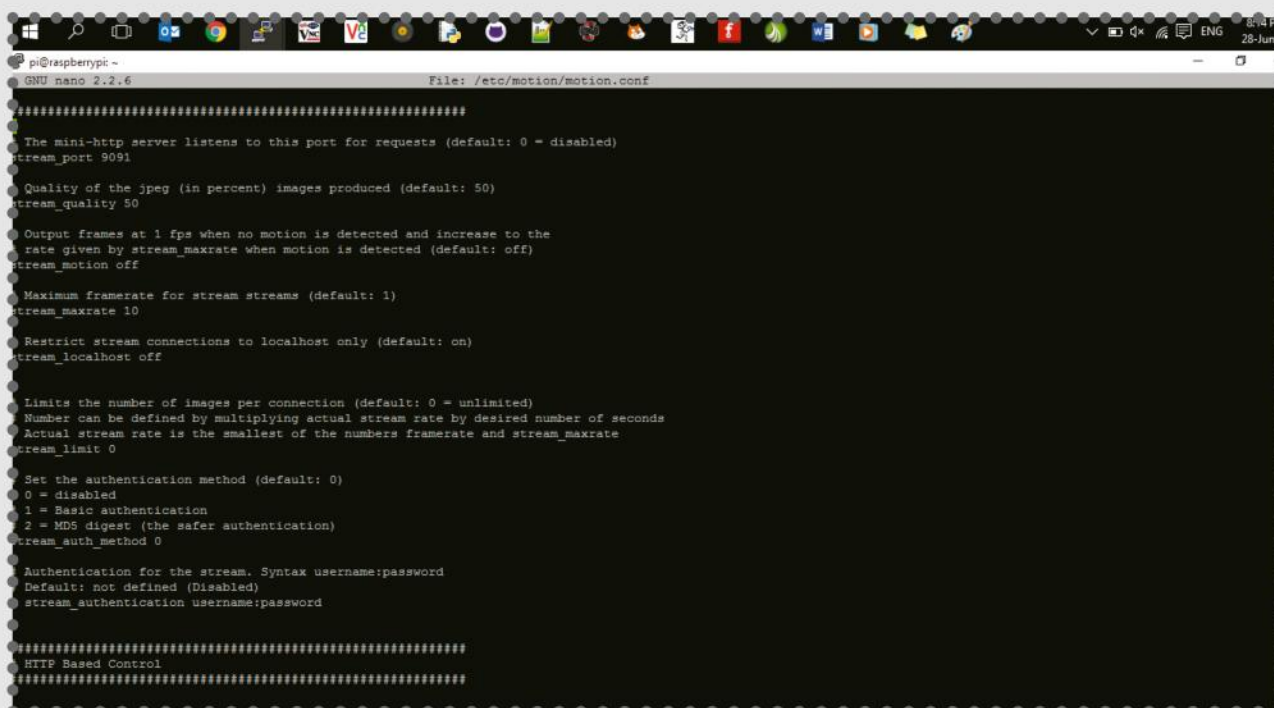
Each time you connect to the internet, your Pi will be given a new IP address; this is called a dynamic IP address. This can cause issues, as when it changes, other devices will no longer be able to locate your Pi. To create one that stays the same (a static IP address), load the LX Terminal, type `ifconfig` and make a note of the `inet addr`, the Broadcast and the Mask numbers. Then type **`route -n`** and note down the Gateway address. Now type **`sudo nano /etc/network/`**



this to OFF so that you can access motion from other computers and devices.

11 Configure the software – part 2

Next, find the `stream_port`, which is set to 8080. This is the port for the video and you can change it if you are having issues viewing the feed; 8081 provides a stable feed. Then, find the `control_localhost` line and set it to OFF. The port that you will access the web config interface is on the `control_port` line and the default is 8080; again, you can change it if you have any streaming issues. The frame rate is the number of frames per second that are captured by the webcam. The higher the frame rate, the better the quality, but setting it higher than 6fps will slow the Pi's performance and produce lag. Finally, set the `post_capture` and specify the number of frames to be captured after motion has been detected.

A screenshot of a terminal window on a Raspberry Pi. The terminal shows the nano text editor editing the file /etc/motion/motion.conf. The file contains configuration settings for the motion software, including stream_port, stream_quality, stream_motion, stream_maxrate, stream_localhost, stream_limit, stream_auth_method, and stream_authentication. The settings are as follows: stream_port 8080, stream_quality 50, stream_motion off, stream_maxrate 10, stream_localhost off, stream_limit 0, stream_auth_method 0, and stream_authentication username:password. The terminal window has a taskbar at the top with various application icons and a system tray on the right showing the date and time as 28-Jun-17, 8:14 PM.

```
pi@raspberrypi ~$ nano /etc/motion/motion.conf
GNU nano 2.2.6 File: /etc/motion/motion.conf

#####

The mini-http server listens to this port for requests (default: 0 = disabled)
stream_port 8080

Quality of the jpeg (in percent) images produced (default: 50)
stream_quality 50

Output frames at 1 fps when no motion is detected and increase to the
rate given by stream_maxrate when motion is detected (default: off)
stream_motion off

Maximum framerate for stream streams (default: 1)
stream_maxrate 10

Restrict stream connections to localhost only (default: on)
stream_localhost off

Limits the number of images per connection (default: 0 = unlimited)
Number can be defined by multiplying actual stream rate by desired number of seconds
Actual stream rate is the smallest of the numbers framerate and stream_maxrate
stream_limit 0

Set the authentication method (default: 0)
0 = disabled
1 = Basic authentication
2 = MD5 digest (the safer authentication)
stream_auth_method 0

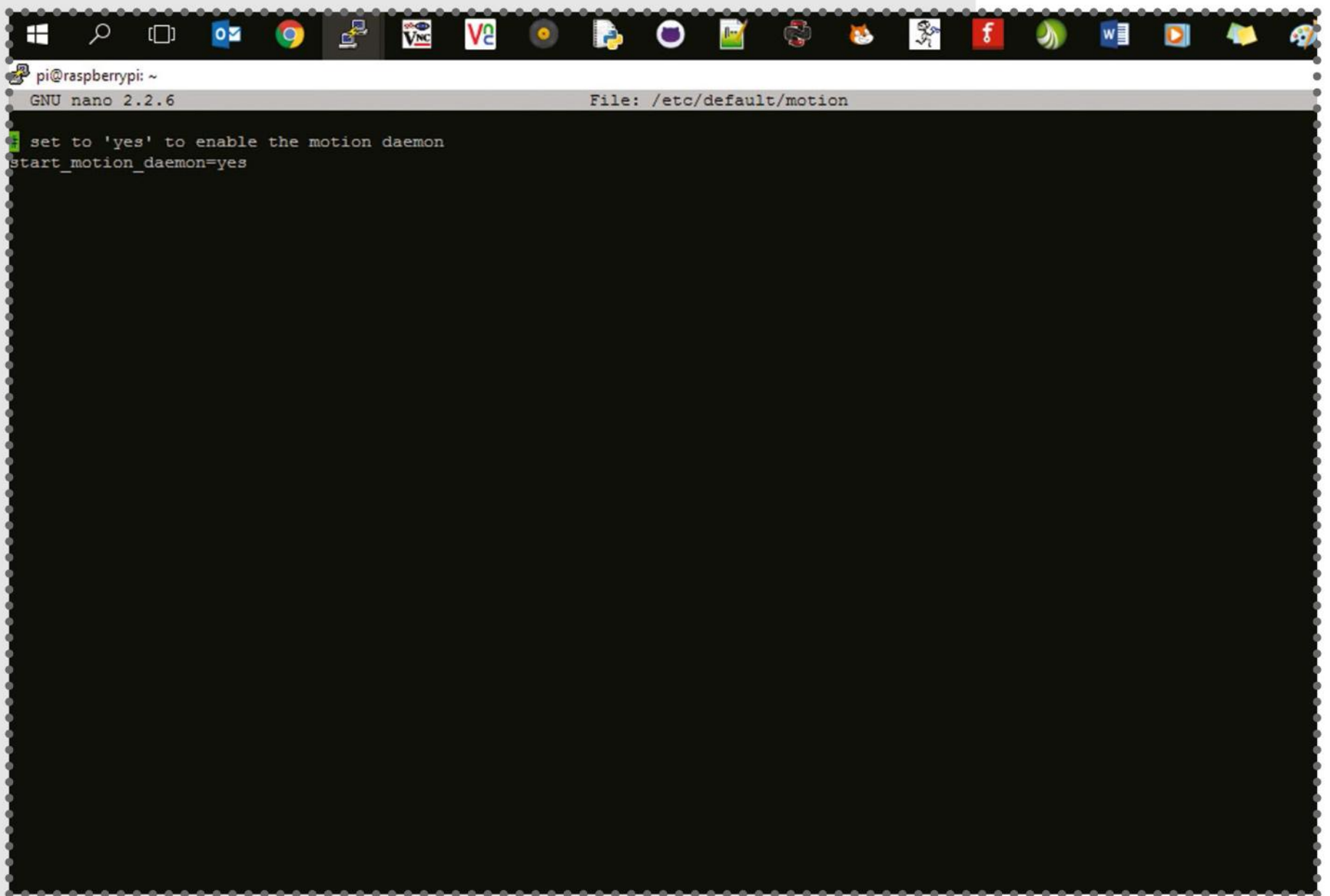
Authentication for the stream. Syntax username:password
Default: not defined (Disabled)
stream_authentication username:password

#####

HTTP Based Control
#####
```

12 Running Motion as a daemon

A daemon is a program that runs in the background



```
pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/default/motion

# set to 'yes' to enable the motion daemon
start_motion_daemon=yes
```

providing a service; in this project you want to run Motion. You do not want to manually have to start it every time you load the Pi; it's better that it automatically starts at bootup. In the LX Terminal, type **sudo nano /etc/default/motion** to edit the file. To set Motion to run as a service from bootup, you need to change the `start_motion_daemon` to Yes:

```
start_motion_daemon=yes
```

Save and exit the file using **CTRL+X** and restart your Pi.

13 Starting the web feed

Before viewing the feed, make a note of your IP



address. In the LX Terminal, type `sudo ifconfig`, although you will have set this in step 8. Then start Motion by typing `sudo service motion start`. Wait for few seconds for it to initiate, then open a browser window on your device. Enter the IP address of your Pi, including the port number that you set in step 11. If you are using VLC player, go to **File>Open Network** and enter the IP address of your Pi followed by the `stream_port` – for example, `192.168.1.50:8080`. The port number in this example is 8080, but you set this in step 11 to 8081 or another value of your choice.

14 Install the Pi radio software

PiFM is a neat little library which enables your Raspberry Pi to broadcast to a radio. Note that this is only for experimentation and should you want to make a public broadcast you must obtain a suitable licence. Getting set up is simple: load the LX Terminal and make a new directory to extract the files into:

```
mkdir PiFM
cd PiFM
```

Then download the required Python files:

```
sudo apt-get update,
sudo apt-get upgrade,
wget http://www.omattos.com/pifm.tar.gz
```

Finally, extract the files using the code:




```
sudo tar xvzf pifm.tar.gz
```

15 Add a simple aerial and broadcast

Setting up the hardware is really easy. In fact, there is no need to attach anything as the Pi can transmit directly from physical pin 7 without the need to alter anything. However, you will probably want to extend the range of the broadcast by adding a wire to GPIO 4 (pin number 7). Unbelievably, this can extend the range of the broadcast to up to 100 metres! Ensure that you are in the PiFM folder and then broadcast the WAV file with the code line:

```
sudo ./pifm name_of_wav_file.wav 100.0
```

In this example, the *Star Wars* theme will play, but you can create and add your own sound files. The '100.0' refers to the FM frequency of the broadcast; this can be changed between a range of 88 and 108MHz. Turn on your radio and adjust to the appropriate frequency and you will hear your message being played.

16 Stop the broadcast

If you wish to end the broadcast before the song or voice has finished, then you will need to kill the transmission. In a new terminal window, type `top`. This will list all the running programs. Look for PiFM somewhere near the beginning of the list and note the ID number. Return to the LX Terminal and type **`sudo kill 2345`**, replacing the 2345 with the appropriate process ID number. This will ensure that each broadcast is new and the Pi is only trying

BCM number

GPIO pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off. You can also program your Raspberry Pi to turn them on or off (output). The GPIO.BCM option means that you are referring to the pins by the 'Broadcom SoC channel' number. The GPIO.BOARD option specifies that you are referring to the pins by their physical location on the header.



to transmit one WAV file at a time.

17 Next time...

You now have four mini hacks which you have prepared and can adapt and combine. You may want to try embedding some of your own hacks that you have created. Next issue, we will cover how to wire up and deploy them!





Check your mail

With Python, you can have your Raspberry Pi act as mail checker, giving you a running list on incoming email



Since the Raspberry Pi is such a small computer, it gets used in a lot of projects where you want to monitor a source of data. One such monitor you might want to create is a mail-checker that can display your unread emails. This issue, we'll look at how to use Python to create your own mail-checking monitor to run on your Pi. We'll focus on the communications between the Pi and the mail server and not worry about how it might be displayed. That will be left as a further exercise.

To start with, most email servers use one of two different communication protocols. The older, simpler one was called POP (Post Office Protocol), and the newer one is called IMAP (Internet Message Access Protocol). We will cover both protocols to cover all of the situations that you might run into. We'll start with the older POP communications protocol. Luckily, there is support for this protocol as part of the standard library. In order to start using it, you will need to import the **poplib** module, and then create a new POP3 object. For example, the following will create a connection to the POP server available through Gmail.


```
import poplib
my_pop = poplib.POP3_
SSL(host='pop.gmail.com')
```

You need to use the POP3_SSL class when connecting to Gmail because Google uses SSL for its connections. If connecting to a different email server, you can use POP3 to make an unencrypted connection. The POP communication protocol involves the client sending a series of commands to the server to interact with it. For example, you can get the welcome message from the server with the `getwelcome()` method:

```
my_pop.getwelcome()
```

The first things that you will want to communicate to the server are the username and password for the email account that you are interested in. Having the username in your code is not too much of a security issue, but the password is another matter. Unless you have a good reason to have it written out in your code, you should probably ask the end-user for it. Included within the standard library is the **getpass** module, which you can use to ask the end-user for their password in a safer fashion. You could use the following code, for example.

```
import getpass
my_pop.user('my_name@
gmail.com')
```

```
my_pop.pass_(getpass.  
getpass())
```

You should now be fully logged in to your email account. Under POP, your account will be locked until you execute the `quit()` method of the connection. If you need a quick summary of what is on the server you can execute the **`stat()`** method:

```
my_pop.stat()
```

This method returns a tuple consisting of the message count and the mailbox size. You can get an explicit list of messages with the **`list()`** method. You have two options for looking at the actual contents of these emails, depending on whether you want to leave the messages untouched or not. If you want to simply look at the first chunk of the messages, you can use the **`top()`** method. The following code will grab the headers and the first five lines of the first message in the list.

```
email_top = my_pop.  
top(1, 5)
```

This method will return a tuple consisting of the response text from the email server, a list of the headers and the number of requested lines, and the octet count for the message. The one problem with the **`top()`** method is that it is not always well implemented on every email server. In those cases, you can use the **`retr()`** method. It will return the entire requested message in the same form as that returned from **`top()`**. Once you have your message contents, you



need to decide what you actually want to display. As an example, you might want to simply print out the subject lines for each message. You could do that with the following code.

```
for line in email_
top[1]:
    if 'Subject' in i:
        print(i)
```

You need to explicitly do the search because the number of lines included in the headers varies from message to message. Once you are done, don't forget to execute the **quit()** method to close down your connection to the email server. One last thing to keep in mind is how long your email server will keep the connection alive. While running test code for this article, it would frequently time out. If you need to, you can use the **noop()** method as a keep-alive for the connection.

As mentioned previously, the second, newer, protocol for talking to email servers is **IMAP**. Luckily, there is a module included in the standard library that you can use, similar to the poplib module we looked at above, called **imaplib**. Also, as above, it contains two main classes to encapsulate the connection details. If you need an SSL connection, you can use **IMAP4_SSL**. Otherwise, you can use **IMAP4** for unencrypted connections. Using Gmail as an example, you can create an SSL connection with the following code.

“Most email servers use one of two communication protocols”


```
import imaplib
import getpass
my_imap = imaplib.
IMAP4_SSL('imap.gmail.com')
```

As opposed to poplib, imaplib has a single method to handle authentication. You can use the getpass module to ask for the password.

```
my_imap.login('my_
username@gmail.com',
getpass.getpass())
```

IMAP contains the concept of a tree of mailboxes where all of your emails are organised. Before you can start to look at the emails, you need to select which mailbox you want to work with. If you don't give a mailbox name, the default is the inbox. This is fine since we only want to display the newest emails which have come in. Most of the interaction methods return a tuple that contains a status flag (either 'OK' or 'NO') and a list containing the actual data. The first thing we need to do after selecting the inbox is to search for all of the messages available, as in the following example.

```
my_imap.select()
typ, email_list = my_
imap.search(None, 'ALL')
```

The **email_list** variable contains a list of binary strings that you can use to fetch individual messages. You should check the value stored in the variable **typ**

to be sure that it contains 'OK'. To loop through the list and select a given email, you can use the following code:

```
for num in email_  
list[0].split():  
    typ, email_raw = my_  
imap.fetch(num, '(RFC822)')
```

The variable **email_raw** contains the entire email body as a single escaped string. While you could parse it to pull out the pieces that you want to display in your email monitor, that kind of defeats the power of Python. Again, available in the standard library is a module called `email` that can handle all of those parsing issues. You will need to import the module in order to use it, as in the example here.

```
import email  
email_mesg = email.  
message_from_bytes(email_  
raw[0][1])
```

All of the sections of your email are now broken down into sections that you can pull out much more easily. Again, to pull out the subject line for a quick display, you can use the code:

```
subject_line = email_  
mesg.get('Subject')
```

There are many different potential items that you could select out. To get the full list of available header

items, you can use the **keys** method, as below:

```
email_mesg.keys()
```

Many times, the emails you get will come as multi-part messages. In these cases, you will need to use the **get_payload()** method to extract any attached parts. It will come back as a list of further email objects. You then need to use the **get_payload()** method on those returned email objects to get the main body. The code might look like:

```
payload1 = email_mesg.  
get_payload()[0]  
body1 = payload1.get_  
payload()
```

As with POP email connections, you may need to do something to keep the connection from timing out. If you do, you can use the **noop()** method of the **IMAP** connection object. This method acts as a keep-alive function. When you are all done, you need to be sure to clean up after yourself before shutting down. The correct way to do this is to close the mailbox first, and then log out from the server. For example:

```
my_imap.logout()  
my_imap.close()
```

You now should have enough information to be able to connect to an email server, get a list of messages, and then pull out the sections that you might want to display as part of your email monitor.





Talking Pi

Join the conversation at...



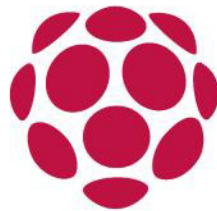
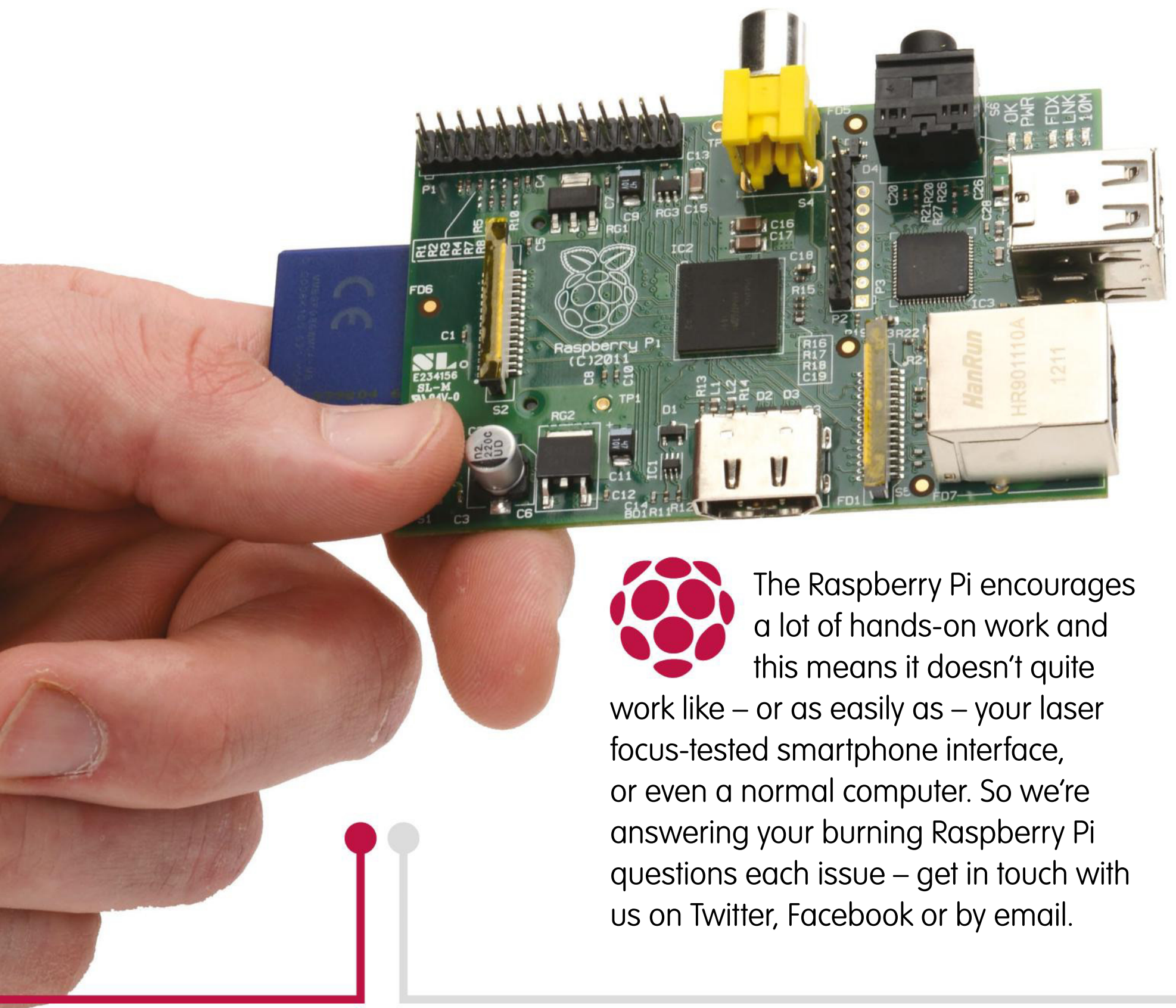
@linuxusermag



Linux User & Developer



linuxuser@futurenet.com

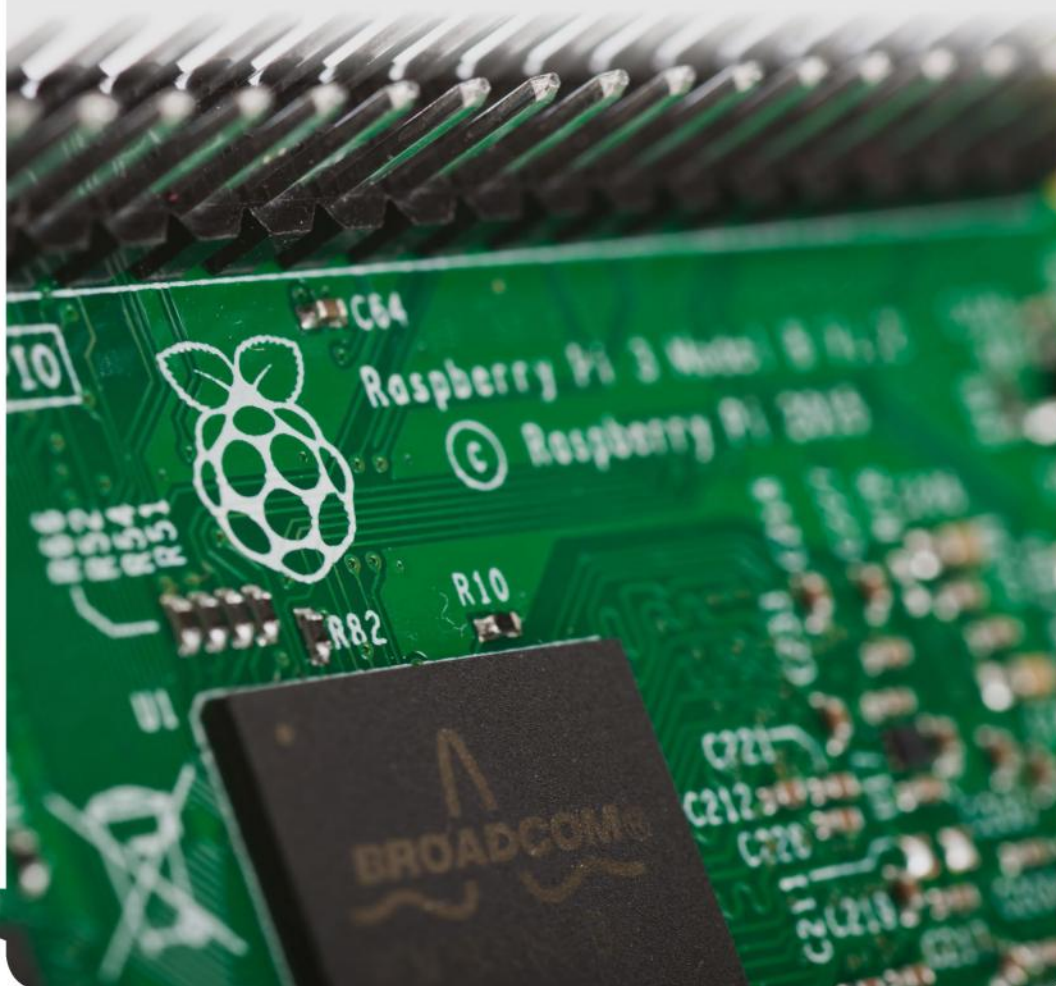


The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easily as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

I've been using a breadboard to practice making circuits and connections for my Pi but I don't think it works!
Ro via email

You haven't said how your breadboard doesn't work, Ro, but we think we might have the solution for you. There are power strips that run through the middle of a breadboard. Sometimes, however, they don't run all the way across but are split in the middle. With some breadboards you can tell fairly

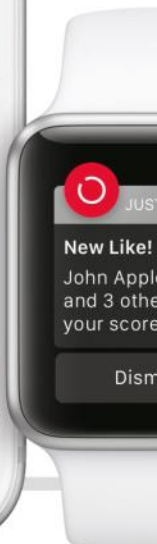
easily, as the blue and red lines that run horizontally along the top have gaps in them in the middle of the board, to show where the power strips stop. But some breadboards, sneakily, don't have this and the lines are continuous, even though the power strips aren't! Try avoiding the holes in the very middle and see if that helps.



Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

JUST A SCORE
WHAT'S YOUR JUST A SCORE?

Have you heard of Just A Score? It's a new, completely free app that gives you all the latest review scores. You can score anything in the world, like and share scores, follow scorers for your favourite topics and much more. And it's really good fun!



I updated my Pi
ages ago and
now Bluetooth
doesn't work...
Jack via email

There was an update a few
months ago that caused some
problems with Bluetooth. Luckily
it's a fairly easy fix, even if
you don't want to do a fresh
installation. Open a terminal

window and enter `sudo apt-get install pi-
bluetooth`, then reboot your Pi. If the Pi says
that you already have the package installed
but it's not working, try `sudo apt-get
--reinstall install pi-bluetooth` and reboot.
It's worth avoiding the use of a touchscreen
during this process, as it can interfere.



My keyboard
isn't behaving
like a regular US
keyboard; why?
Shawn via email

That'd be because Raspbian
and NOOBS default to UK
English, which has a slightly
different keyboard layout (Don't
ask us why, it's annoyed Brits
for decades). It's easy enough

for y'all to fix (*Stop that immediately – Ed*)
– open up a terminal in Raspbian and
type `sudo raspi-config`, then choose
Internationalization>Keyboard Setup. If your
keyboard isn't listed just choose a generic
one (101, 102 or 104), and scroll down to
Other. Under the Country of Origin menu,
select English (US). Complete everything
else it asks you to, then reboot your Pi. You
should find everything where you expect it.



**JUSTA
SCORE**
WHAT'S YOUR **JUST A SCORE?**

You can score
absolutely anything on
Just A Score. We love
to keep an eye on free/
libre software to see
what you think is worth
downloading...

10 Keybase

9 Cinnamon Desktop

8 Tomahawk

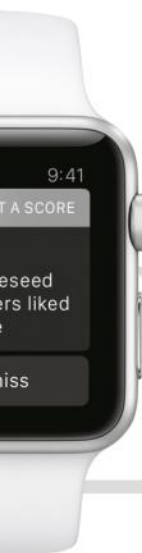
4 Anaconda installer

3 FOSS That Hasn't Been
Maintained In Years

SCORE ANYTHING
JUST A SCORE



Download on the
App Store

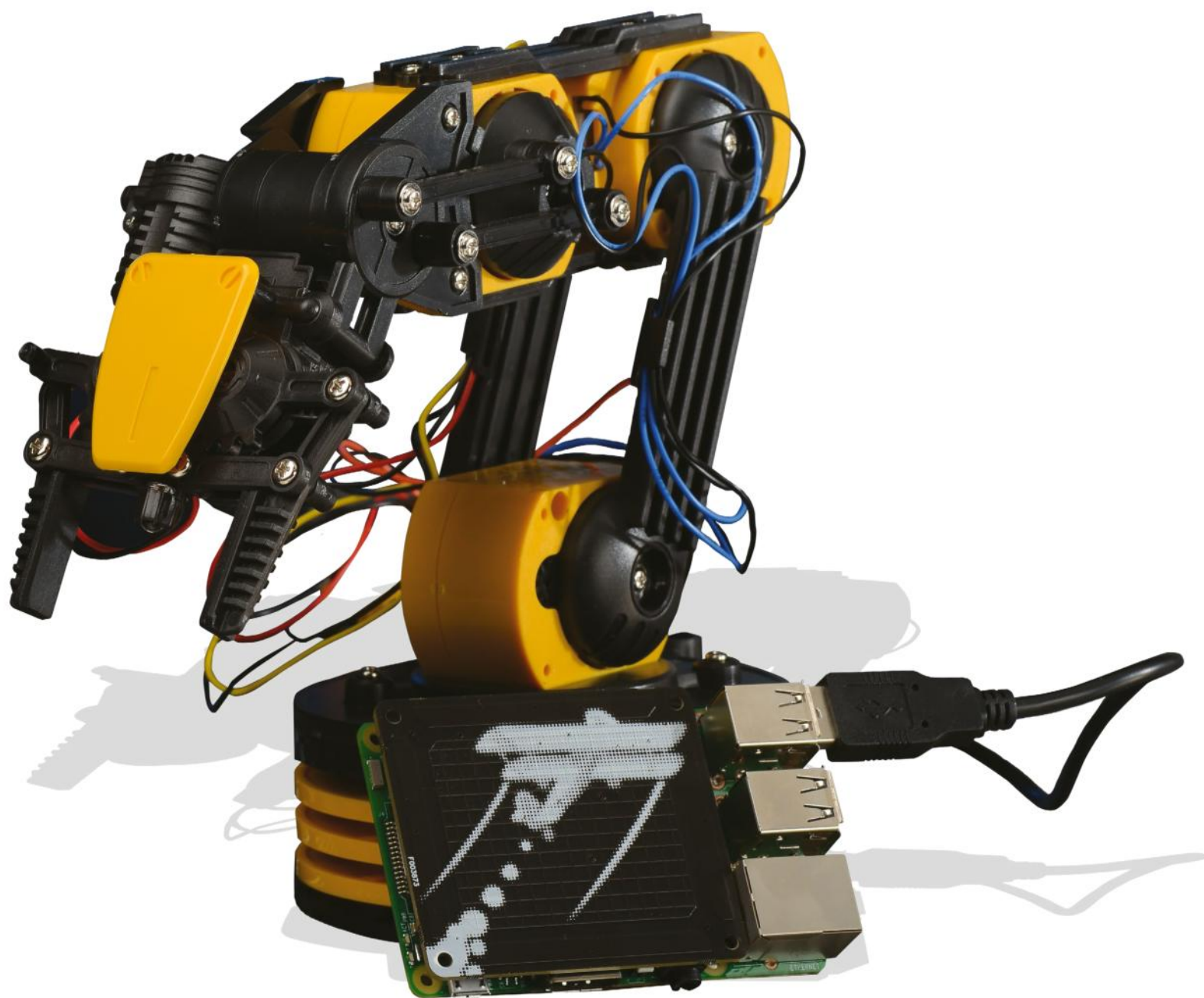




Next issue

Get inspired Expert advice Easy-to-follow guides

Get a helping hand



Get this issue's source code at:
www.linuxuser.co.uk/raspicode

FOR THE GNU GENERATION



5 issues for £5!

www.imaginesubs.co.uk/spring171

BUY YOUR ISSUE TODAY

Available from all good newsagents & supermarkets today

TERMS AND CONDITIONS The trial offer is for new UK print subscribers paying by Direct Debit only. You will receive 13 issues in a year. Full details of the Direct Debit guarantee are available upon request. If you are dissatisfied in any way you can write to us or call us to cancel your subscription at any time and we will refund you for all un-mailed issues. Prices correct at point of print and subject to change. Offer ends 31/05/2017.